10.09.03

日本国特許庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出願年月日
Date of Application: ２００２年　９月１０日

出願番号
Application Number: 特願２００２－２６４２８３

[ST. 10/C]: [ＪＰ２００２－２６４２８３]

出願人
Applicant(s): 玉津　雅晴
アネックスシステムズ株式会社

２００３年１０月１７日

特許庁長官
Commissioner,
Japan Patent Office

今井康夫

【書類名】　特許願
【提出日】　平成14年 9月10日
【整理番号】　A-11-7
【あて先】　特許庁長官殿
【発明者】
　【住所又は居所】　東京都多摩市馬引沢２丁目１４番１４号　サンセットヒルズ２
　【氏名】　玉津　雅晴
【特許出願人】
　【識別番号】　592159324
　【氏名又は名称】　玉津　雅晴
【特許出願人】
　【識別番号】　593050596
　【氏名又は名称】　アネックスシステムズ株式会社
【代理人】
　【識別番号】　100076141
　【弁理士】
　【氏名又は名称】　市之瀬　宮夫
【手数料の表示】
　【予納台帳番号】　062156
　【納付金額】　21,000円
【提出物件の目録】
　【物件名】　明細書　1
　【物件名】　図面　1
　【物件名】　要約書　1
　【包括委任状番号】　0106954
　【包括委任状番号】　0106934
【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　データ及びデータベースの無停止自動再編成システム、並びに、データ及びデータベース

【特許請求の範囲】

【請求項１】　一つのユニークな主キーとゼロ個または１個以上のノンユニークな代替キーを持つレコードを順次格納するブロックと、前記ブロックの位置管理を、当該ブロックとランダム・アクセス・メモリ上のアドレスとを対応させたロケーション・テーブルを用いて行い、前記ランダム・アクセス・メモリに格納されているデータ及びデータベースを管理するコンピューターによるシステムであって、

データベース再編成指令を受けると、前記現用のロケーション・テーブルに加えて新規のロケーション・テーブルを作成する第１の手段と、

単位処理時点で一個または複数個のブロックを対象とし、現用のロケーション・テーブルのエントリーを新規のロケーション・テーブルに順次、書き移してゆき順次、書き移してゆく際に、オーバーフロー・ブロックを検出したときには、当該オーバーフロー・ブロックの連結を切り離し、新規のロケーション・テーブルに新しいエントリーを追加し、新規のロケーション・テーブルのプライマリー・ブロックとする第２の手段と、

を備えたことを特徴とするデータ及びデータベースの無停止自動再編成システム。

【請求項２】　一つのユニークな主キーとゼロ個または１個以上のノンユニークな代替キーを持つレコードを順次格納するブロックと、前記ブロックの位置管理を、当該ブロックとランダム・アクセス・メモリ上のアドレスとを対応させたロケーション・テーブルを用いて行い、前記ランダム・アクセス・メモリに格納されているデータ及びデータベースを管理するコンピューターによるシステムであって、

前記ブロック内の格納率が所定の値の範囲外である場合に前後のブロックのレコードを移動してフラグメンテーションを解消する手段と、

を備えたことを特徴とするデータ及びデータベースの無停止自動再編成システ

ム。

【請求項３】　前記現用のロケーション・テーブル及び前記新規のロケーション・テーブルにそれぞれ再編成ポインターを設け、各再編成ポインターに対して、単位処理時点で一個または複数個のブロックを対象として順次前記再編成処理が終了した位置を格納し、再編成がラスト・ポインターまで到達したときに再編成処理を完了する手段を備えたことを特徴とする請求項１又は２記載のデータベース無停止自動再編成システム。

【請求項４】　再編成をしている最中に、レコードを主キーで検索する場合には、目的の主キー値が再編成ポインターの指しているエントリーのプライマリー・ブロック及びオーバーフロー・ブロックに含まれるレコードの主キー値より大きいか小さいかを比較する比較手段と、

前記比較手段により目的の主キー値が、再編成ポインターが指しているブロックに格納されているレコードの主キー値以上と判断されたときには現用のロケーション・テーブルを使用して目的のレコードを検索し、主キー値未満と判断されたときには新規のロケーション・テーブルを使用して目的のレコードを検索する検索手段と、

を備えたことを特徴とするデータ及びデータベースの無停止自動再編成システム。

【請求項５】 代替キーと当該代替キー値のレコードが格納されているブロック番号と当該レコードの主キーとからなるエントリーを代替キーの順番に並べて複数格納可能とするとともに、予め同一の大きさで必要数を連続して確保できる代替キー・ブロックを使用し、代替キー・テーブルのエントリーは代替キー・ブロック中に代替キーの順番で並ぶように格納されており、同一の代替キーを持つ代替キー・テーブルのエントリーは同一の代替キー・ブロックに格納されており、同一の代替キーを持つエントリーの数が多いかもしくは代替キーの挿入により、代替キー・ブロックに格納できないときに、代替キー・オーバーフロー・ブロックを代替キー・ブロックに追加してエントリーを格納するデータ格納検索システムであって、

データベース再編成指令を受けると、前記現用代替キー・テーブルに加えて現

用代替キー・テーブルを作成する第１の手段と、

単位処理時点で１個または複数個のブロックを対象とし、現用代替キー・テーブルのエントリーを現用代替キー・テーブルに順次、書き移してゆき、

順次、書き移してゆく際に、代替キー・ブロックに連結されている代替キー・オーバーフロー・ブロックを検出したときには、当該代替キー・オーバーフロー・ブロックの連結を切り離し、新たな代替キー・ブロックとして現用代替キー・テーブルに書き移す第２の手段と、

を備えたことを特徴とするデータ及びデータベースの無停止自動再編成システム。

【請求項６】　代替キーと当該代替キー値のレコードが格納されているブロック番号と当該レコードの主キーとからなるエントリーを代替キーの順番に並べて複数格納可能とするとともに、予め同一の大きさで必要数を連続して確保できる代替キー・ブロックを使用し、代替キー・テーブルのエントリーは代替キー・ブロック中に代替キーの順番で並ぶように格納されており、同一の代替キーを持つ代替キー・テーブルのエントリーは同一の代替キー・ブロックに格納されており、同一の代替キーを持つエントリーの数が多いかもしくは代替キーの挿入により、代替キー・ブロックに格納できないときに、代替キー・オーバーフロー・ブロックを代替キー・ブロックに追加してエントリーを格納するデータ検索格納システムであって、

前記代替キー・ブロック内の格納率が所定の値の範囲外である場合に前後の代替キー・ブロックのレコードを移動してフラグメンテーションを解消する手段と、

を備えたことを特徴とするデータ及びデータベースの無停止自動再編成システム。

【請求項７】　前記現用代替キー・テーブル及び前記現用代替キー・テーブルにそれぞれ再編成ポインターを設け、各再編成ポインターに対して、単位処理時点で一個または複数個のブロックを対象として順次前記再編成処理が終了した位置を格納し、再編成がラスト・ポインターまで到達したときに再編成処理を完了する手段を備えたことを特徴とする請求項５又は６記載のデータ及びデータベ

ースの無停止自動再編成システム。

【請求項８】　再編成をしている最中に、レコードを、代替キーで検索する場合に、目的の代替キー値が再編成ポインターの指しているエントリーの代替キー・ブロックに含まれるエントリーの代替キー値より大きいか小さいかを比較する比較手段と、

前記比較手段により目的の代替キー値が再編成ポインターの指している代替キー・ブロックに格納されているエントリーの代替キー値以上と判断されたときには現用代替キー・テーブルを使用して目的のエントリーを検索し、前記エントリーの代替キー値未満と判断されたときには現用代替キー・テーブルを使用して目的のレコードを検索する検索手段と、

を備えたことを特徴とするデータ及びデータベースの無停止自動再編成システム。

【請求項９】　代替キーと当該代替キー値のレコードが格納されているブロック番号と当該レコードの主キーとからなるエントリーを代替キーの順番に並べて複数格納可能とするとともに、予め同一の大きさで必要数を連続して確保できる代替キー・ブロックを使用し、代替キー・テーブルのエントリーは代替キー・ブロック中に代替キーの順番で並ぶように格納されており、同一の代替キーを持つ代替キー・テーブルのエントリーは同一の代替キー・ブロックに格納されており、同一の代替キーを持つエントリーの数が多いかもしくは代替キーの挿入により、代替キー・ブロックに格納できないときに、代替キー・オーバーフロー・ブロックを代替キー・ブロックに追加してエントリーを格納するデータ検索格納システムであって

データベース再編成指令を受けると、前記現用代替キー・ロケーション・テーブルに対して新規に代替キー・ロケーション・テーブルを作成する第１の手段と、

単位処理時点で1個または複数個のブロックを対象とし、現用代替キー・ロケーション・テーブルのエントリーを新規の代替キー・ロケーション・テーブルに順次、書き移してゆき、

順次、書き移してゆく際に、代替キー・ブロックに連結されている代替キー・

オーバーフロー・ブロックを検出したときには、当該代替キー・オーバーフロー・ブロックの連結を切り離し、新規の代替キー・ロケーション・テーブルに新しいエントリーを追加し、新規の代替キー・ロケーション・テーブルの代替キー・ブロックとする第２の手段と、

を備えたことを特徴とするデータ及びデータベースの無停止自動再編成システム。

【請求項１０】　代替キーと当該代替キー値のレコードが格納されているブロック番号と当該レコードの主キーとからなるエントリーを代替キーの順番に並べて複数格納可能とするとともに、予め同一の大きさで必要数を連続して確保できる代替キー・ブロックを使用し、代替キー・テーブルのエントリーは代替キー・ブロック中に代替キーの順番で並ぶように格納されており、同一の代替キーを持つ代替キー・テーブルのエントリーは同一の代替キー・ブロックに格納されており、同一の代替キーを持つエントリーの数が多いかもしくは代替キーの挿入により、代替キー・ブロックに格納できないときに、代替キー・オーバーフロー・ブロックを代替キー・ブロックに追加してエントリーを格納するデータ検索格納システムであって、

前記代替キー・ブロック内の格納率が所定の値の範囲外である場合に前後の代替キー・ブロックのレコードを移動してフラグメンテーションを解消する手段、

を備えたことを特徴とするデータ及びデータベースの無停止自動再編成システム。

【請求項１１】　前記現用代替キー・ロケーション・テーブル及び前記新規の代替キー・ロケーション・テーブルにそれぞれ再編成ポインターを設け、前記各再編成ポインターに単位処理時点での再編成が終わっている位置を格納する手段を備えたことを特徴とする請求項９又は１０記載のデータベース無停止自動再編成システム。

【請求項１２】　再編成をしている最中に、レコードを、再編成を行っている代替キーで 検索する場合には、目的の代替キー値が再編成ポインターの指しているエントリーの代替キーブロックに含まれるエントリーの代替キー値より大きいか小さいかを比較する比較手段と、

前記比較手段により目的の代替キー値が再編成ポインターの指している代替キー・ブロックに格納されているエントリーの代替キー値以上と判断されたときには現用代替キー・ロケーション・テーブルを使用して目的のレコードを検索し、前記エントリーの代替キー値未満と判断されたときには新規の代替キー・ロケーション・テーブルを使用して目的のレコードを検索する検索手段と、

を備えたことを特徴とするデータ及びデータベースの無停止自動再編成システム。

【請求項１３】　コンピューターにおけるデータおよびデータベース・システムにおいて、

代替キーと該当レコードが格納されているブロック番号と該当レコードの主キーとからなるエントリーを複数格納可能とするとともに、予め同一大きさで必要数を連続して確保できる代替キー・ブロックを使用し、

前記代替キー・ブロックの記憶装置内での位置を管理するため、前記代替キー・ブロックに付与した番号と前記記憶装置内の物理的位置を対応させた代替キー・ロケーション・テーブルとを用い、

前記キー・ブロックに代替キー･エントリーを代替キーの順番で格納するとともに、前記代替キー・ブロックに格納できなくなったときに、新たな代替キー・オーバーフロー・ブロックを割り当てて代替キー･エントリーを格納し、かつ、前記代替キー・ロケーション・テーブルにより前記代替キー・ブロックの前記記憶装置内での位置管理を行うことを特徴とするデータおよびデータベース・システム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明はコンピューターによるデータ及びデータベースの格納検索システムに係り、特にシステムの運用中に無停止で自動的にデータ及びデータベースの再編成を行うことができるようにしたデータ及びデータベースの無停止自動再編成システムに関するものである。

【０００２】

【従来の技術】

従来のコンピューターによるデータベース格納検索方式は、「Jeffrey D.Ullman著、国井他１名訳、”データベース・システムの原理”，第1版，日本コンピューター協会，1985年5月25日,p45-71」、「Samuel Leffler他著/中村明他訳，”UNIX（登録商標）4.3BSDの設計と実装”,丸善株式会社，1991年6月30日,p193-191」及び「Michael J.Folk他著/楠本博之訳,”bit別冊 ファイル構造”,共立出版 株式会社,1997年6月5日,p169-191」に記載されているように、基本的には階層型のインデックスを使用するものであった。

【０００３】

このような 従来のデータベース格納検索方式によれば、

（１）インデックスの創生や維持に負荷がかかること、

（２）最終的に使用されると想定される最大のブロックを予め生成しておかなければならないこと、

（３）インデックスが階層構造をとっているため、データの挿入や削除によってインデックスの更新が行われる場合に、上位インデックスまで変更される場合が発生するために排他範囲が広くなり、デッドロックを引き起こしやすいこと、

などの不都合があった。

【０００４】

このような従来のデータベース格納検索方式の不都合を解消するため、本発明者は、従来の階層型インデックスに代えてロケーション・テーブルと代替キー・テーブルという概念を導入し、インデックスの処理に伴う複雑な処理を簡素化し、テーブル自体の検索をバイナリー・サーチなどの手法を用いることにより、高速化と、メンテナンスの容易性を確保できるようにしたデータ格納検索方式を提案した（特開平１１－２３１０９６号公報、米国特許第６４１５３７５号参照）。

【０００５】

ここで、本発明者が提案したデータ格納検索方式の内容について簡単に説明しておくことにする。本発明のデータ格納検索方式は、ロケーション・テーブルと代替キー・テーブルを使用し、それらに対して、バイナリー・サーチを行うこと

により、目的のレコードを検索するものである。レコードはブロックと言う固定長の格納領域に格納する。レコードは、当初はプライマリー・ブロックに格納するが、そのプライマリー・ブロックが満杯になった後、そのプライマリー・ブロックに対して、挿入レコードが発生するとレコードを格納できなくなるため、当該プライマリー・ブロックに対して、オーバーフロー・ブロックを作成し連結し、一部のレコードをオーバーフロー・ブロックに移動した後、プライマリー・ブロックにレコードの格納を行う。そのオーバーフロー・ブロックが満杯になった後で、更にレコードの挿入が発生すると、そのオーバーフロー・ブロックに対して、オーバーフロー・ブロックを作成して連結する。

【０００６】

ここで、連結とは、物理的に連結されていることを意味するのではなく、プライマリー・ブロックが一番目のオーバーフロー・ブロックのアドレスを保持し、１番目のオーバーフロー・ブロックが２番目のオーバーフロー・ブロックのアドレスを保持している状態が、あたかも物理的にブロックが繋がれているように扱えることから、そのような表現を使用している（以下、同様である）。

このように本発明者の提案したデータ格納検索方式では、オーバーフロー・ブロックは無制限に連結できるので、レコードが格納できなくなる状態は発生しないという利点がある。

【０００７】

【発明が解決しようとする課題】

しかしながら、本発明が提案したデータ格納検索方式によれば、次のような問題点が発生する。

（１）上述したようにオーバーフロー・ブロックが多数連結されると、ロケーション・テーブルのエントリーを検索して、プライマリー・ブロックを検出した後、目的のレコードを探すのに、プライマリー・ブロックしか存在しない場合に対して、時間が余分に掛かることになる。

（２）また、レコードは、ブロック中に主キー値の順番に格納することになっており、これは、プライマリー・ブロックに連結された、オーバーフロー・ブロック内や、プライマリー・ブロックとオーバーフロー・ブロック間でも同様であ

る。このように主キー値の順番に格納するので、レコードを挿入する場合に、多くのオーバーフロー・ブロックが連結されていると、複数のオーバーフロー・ブロックにまたがって、レコードを移動する必要がある場合が発生し、プライマリー・ブロックしか存在しない場合に比較して、時間が余分に掛ることになる。

（３）この他、オーバーフロー・ブロックが発生した後に、レコードの挿入があまり発生しない場合や、データの削除によりブロック内のレコードが減少した場合には、当該プライマリー・ブロックやオーバーフロー・ブロック内に空き領域が発生するが、そのブロックにレコードの挿入が行われないと、ブロック内の空スペースは、使用されないまま残ることになり、格納領域の格納効率が落ちることもあった。

（４）同様に、代替キー・テーブルのエントリーは、通常、挿入型で発生するため、代替キー・ブロックに対して、代替キー・オーバーフロー・ブロックが発生しやすく、代替キー・オーバーフロー・ブロックが存在しない場合に比較して、代替キーによるアクセス速度が低下する、という問題点を持っていた。これは、ターゲット・キー値に対して、目的の代替キー・ブロックを捜しても、その代替キー・ブロックにオーバーフロー・ブロックが連結されていると、それらを検索して目的のエントリーを探さなければならないからである。

（５）更に、ブロックに関して述べたのと同様に、エントリーを挿入する際にも、エントリーは代替キー・ブロック内で、代替キー値の順番に格納され、その順序は、代替キー・オーバーフロー・ブロックが存在するときには、代替キー・ブロックと代替キー・オーバーフロー・ブロックの相互間でも、順番に並ぶようになっている。このため、代替キー・ブロックしか存在しない場合に比較して、目的のエントリーを検索するのに余分な時間がかかるものであった。他に、代替キー・エントリーの挿入では、エントリーの移動が必要になり、代替キー・オーバーフロー・ブロックが多い場合には、移動量が多くなるという欠点もあった。

（６）加えて、代替キー・テーブルに関しては、キーの発生が追加型ではなく、挿入型になることは前述したが、こうした挿入に対応するための工夫として、本発明者が提案したデータ格納検索方式では、プレ代替キー・テーブルを使用してエントリーの少ない状態で、代替キー・オーバーフロー・ブロックの発生を吸

収する方式も併せて考案されているが、完全にオーバーフロー・ブロックを無くすことは困難であった。

【０００８】

ところで、上述した（１）～（６）で説明した問題は、従来の階層型のインデックスを使用したデータ格納検索方式の方が顕著であって、その従来の階層型のインデックスを使用したデータ格納検索方式では、データの挿入によってインデックスが分裂していくために、インデックスの形態が変化し、アクセス・スピードの鈍化を招くという欠点があった。

このような従来の階層型のインデックスを使用したデータ格納検索方式の欠点を解消するためには、再創生または再編成と呼ばれる方法を使用する必要がある。

【０００９】

この再創生または再編成と呼ばれる方法は、次のようにしていた。すなわち、一旦システムを全面的に停止し、システム上に記録されているレコード（データ）を、すべて別の媒体にコピーする。ついで、元のデータ（レコード、主キーのインデックス、代替キーのインデックス）を消去し、レコードの書き込みを行う。次に、主キーのインデックス、代替キーのインデックスの作成を行う。主キーのインデックスの作成は、総てのレコードから主キーを読み出し、レコードの格納されているブロックのアドレスなどと組にしたエントリーを作成し、その後、主キーでソートを行い、最下位層のインデックスを作成し、その後、上位のインデックスを順次作成する、というものであった。

【００１０】

代替キーのインデックスの作成も同様で、レコードから代替キーを読み出し、レコードの格納されているブロックのアドレスなどと組にしたエントリーを作成し、その後、代替キーでソートを行い、最下位層のインデックスを作成し、その後、上位のインデックスを順次作成する、というものであった。これは、代替キーの種類が複数ある場合には、それ毎に実行する必要があった。

このような順序で作業を行うため、データ量やインデックスの種類（数）等によってケース・バイ・ケースであるが、従来の階層型のインデックスを使用した

データ格納検索方式においては、再創生に数時間から数百時間という長い時間が必要で、その間、システムを使うことができない、という欠点があった。

また、上記のように面倒な方法を何段階も実行しなければならないため、再創生を全く自動で、人手を掛けずに実施することが困難であり、システム関係者が夜間や休日などに再創生を実施する、といった、勤務上で苦痛を伴う問題もあった。

【００１１】

更に、２４時間無停止システムでは、アクセス効率が低下してきても、再創生を実施することができず、性能の劣化をやむなく受け入れ、高性能のハードウエアで性能劣化をカバーするなど、費用的に負担がかかるものでもあった。

特に最近では、ハードウエア自体が二重化するなどして、無停止運転が可能になってきており、データベースの無停止運転が困難であるのは問題となってきているものであった。

【００１２】

これに対して、本発明者が提案したデータ格納検索方式における再創生は、基本的には、従来の階層型のインデックスを使用したデータ格納検索方式と同等であるが、本発明者が提案したデータ格納検索方式では、従来の階層型のインデックスを使用したデータ格納検索方式のように、複雑なインデックスを使用しておらず、ロケーション・テーブルと代替キー・テーブルという構成になっているため、従来の階層型のインデックスを使用したデータ格納検索方式に比較して、時間的には遥かに短時間で再創生が完了するが、再創生中にデータベースが使用できないことは同様であった。

【００１３】

具体的には、本発明が提案したデータ格納検索方式における再創生は、プライマリー・システムの運用を止めて、ブロックに格納されているレコードを読み取り、別の装置に格納し、その後で、プライマリー・システムのロケーション・テーブル、ブロック、代替キー種別毎の代替キー・テーブルを再作成し、別の装置に格納したレコードを、新たに作製したブロックに格納しながらロケーション・テーブルのエントリーを作成し、データの格納が終了してから、種類毎の代替キ

ーの代替キー・テーブルのエントリーを作成するという方法である。この方法では、従来のデータ格納検索方式と同等で、プライマリー・システムの運用を止める必要があるため、無停止システムには適用が困難であった。

【００１４】

本発明は、上述した点に鑑みてなされたもので、システムの運用中に無停止で自動的にデータ及びデータベースの再編成を行うことができるようにしたデータ及びデータベースの無停止自動再編成システム、並びに、データ及びデータベースを提供することを目的とする。

【００１５】

【課題を解決しようとする手段】

上記目的を達成するため、本発明では、次のようにしている。

（本発明の基礎）

まず、本発明者が提案したデータ格納検索方式では、ロケーション・テーブルと代替キー・テーブルを使用し、それに対して、バイナリー・サーチを行うことにより、目的のレコードを検索している。このレコードは、ブロックと言う固定長の格納領域に格納する。このブロックは、プライマリー・ブロックのみがロケーション・テーブルによって管理されており、オーバーフロー・ブロックはプライマリー・ブロックによって管理されている。ロケーション・テーブル及び代替キー・テーブルは、双方とも階層構造を持たないフラットなテーブルである。

【００１６】

本発明者の提案したデータ格納検索方式が有するこの特徴を活かして、ロケーション・テーブルとブロックの再編成を行うこととする。また、代替キー・テーブルに関しても同様に再編成を行うこととするである。

ここで、再編成とは、システム上に格納されたデータに対して、データの挿入・追加・変更・削除が行われるために、データの構成が変動し、データのアクセスが遅くなるため、データやインデックスを整頓することであり、（ｉ）オーバーフロー・ブロックの解消、（ii）フラグメンテーションの解消、（iii）適正な初期格納率の確保の３つを行うことである。以下に、前記（ｉ）項ないし（iii）項の内容について説明することにする。

【００１７】

（ｉ）オーバーフロー・ブロックの解消とは、次のようなことである。オーバーフロー・ブロックは、レコードの挿入によって発生する。すなわち、あるプライマリー・ブロックにレコードが満杯に入っており、更に、そのプライマリー・ブロックに対して、レコードを挿入しようとする場合に、そのままでは挿入できないことになる。このような挿入を可能とするために、当該プライマリー・ブロックに対して、オーバーフロー・ブロックを割り当て、必要な数のレコードを、プライマリー・ブロックからオーバーフロー・ブロックに移動し、元のプライマリー・ブロックに目的のレコードを挿入することにより、レコードを格納することが可能となる。

ところが、オーバーフロー・ブロックが存在すると、プライマリー・ブロックのみの場合に比較して、レコード検索の負荷が余分に必要となるため、オーバーフロー・ブロックをプライマリー・ブロックとして、ロケーション・テーブルで管理するようにすることが、早いアクセスを実現する上で、必要なことになる。

また、レコードをブロック内で主キーの順番に並べるために、レコードの挿入ではレコードの移動が必要になるが、オーバーフロー・ブロックが多いと、移動対象レコードの数が増加し、効率上の問題になるものであった。

【００１８】

（ii）フラグメンテーションの解消とは次のようなことである。フラグメンテーションとは、格納領域の分散化である。ブロック（プライマリー・ブロック及びオーバーフロー・ブロック）に格納されているレコードが不要になって、消去するとその分、ブロックの格納領域が空くことになる。挿入されるレコードが無い場合には、格納領域が空いたまま無駄になってしまう。また、オーバーフロー・ブロックが発生して、当該オーバーフロー・ブロックの格納容量に比較して、少数のレコードが格納された状態で、レコード挿入がその後発生しないと、その空き容量も、使用されない無駄な格納領域になってしまう。

このような無駄な格納領域を無くすために、次のブロックに格納されているレコードを前のブロックに移動して、ブロック内にレコードが十分に格納されるようにすることにより、格納領域を使用する上での無駄が解消できる。

【００１９】

(iii)適正な初期格納率の確保に関して説明する。初期格納率とは、最初にブロックを作成してレコードを書き込む場合に、予め一定割合のブロック内領域を空けておき、オーバーフロー・ブロックの発生をある程度防止するためのものである。

初期に、ブロックにレコードを格納する場合に、ブロックの容量の１００％に対してレコードを格納することは可能であるが、そのように満杯になるまでレコードを格納した状態で、レコードが挿入されると、直ちにオーバーフロー・ブロックが発生することになる。このようになることを避けるために、例えば、当初、プライマリー・ブロックにレコードを格納する場合に、格納容量の９０％までを限度としてレコードを格納しておき、その後、挿入レコードが発生した場合に、その空領域を使用してレコードの格納を行うことにより、直ちにオーバーフロー・ブロックを発生させないようにしておくことが可能である。

以上では、ロケーション・テーブルとブロック、オーバーフロー・ブロックに対する考え方を中心に記述したが、代替キー・ブロック、代替キー・オーバーフロー・ブロックに対しても同様である。

【００２０】

（第１の発明）

本第１の発明は、ロケーション・テーブルとブロックの再編成に係るものであり、以下に説明する。

本第１の発明では、ロケーション・テーブルのエントリーは、そのエントリーが指しているブロックの番号、ブロックのアドレスの他に、必要に応じて、そのエントリーが指しているブロック及び、そのブロックに連結されているオーバーフロー・ブロックすべてに格納されているレコードの、主キー値の最小値と最大値の双方、または、何れか一方を保持している点に着目し、次のようにしている。

【００２１】

すなわち、本第１の発明では、ロケーション・テーブルに対して、新規にロケーション・テーブルを作成し、現用のロケーション・テーブルのエントリーから

新規のロケーション・テーブルに順次、書き移してゆく。本発明において、「書き移す」とは、情報をそのまま複写する場合と、必要に応じて一部の情報を変更して、その変更した情報を書くことの両方を言う。本発明では、この順次書き移してゆく際に、プライマリー・ブロックに連結されているオーバーフロー・ブロックの、連結を切り離し、新規のロケーション・テーブルに新しいエントリーを追加し、当該オーバーフロー・ブロックを移動することなく、新規のロケーション・テーブルのプライマリー・ブロックとする。本第１の発明では、このようにしたことにより、オーバーフロー・ブロックの解消を行っている。

【００２２】

本第１の発明では、フラグメンテーションの解消を次のようにしている。すなわち、フラグメンテーションの解消は、複数のブロック（プライマリー・ブロック及びオーバーフロー・ブロック）の格納率を調べて、複数のブロックに対して、ブロック相互間でレコードの移動を行い、必要に応じて、新規にブロックを追加しロケーション・テーブルのエシトリーから追加するか、使用されていたブロックを未使用にしロケーション・テーブルのエントリーから削除する、ことにより実現される。上記の動作は、１個または複数個のブロックを対象にして、排他を掛けた状態で実施する。

適正な初期格納率の確保は、フラグメンテーションの解消と似ており、ブロックの中にレコードが占める容量を、定めた初期格納率とするように、レコードの移動を行うことにより実現される。

【００２３】

以上説明したことを総合すると、単位処理時点（以下、「一時点」ということもある。）、１個または複数個のブロックに対して排他を掛けて、再編成を実施する。それは、オーバーフロー・ブロックの解消、フラグメンテーションの解消、適正な初期格納率の確保の３項目を同時に実施することである。当該ブロックに対する再編成が完了したら、排他を解除し、当該ブロックを使用できる状態にする。上記の再編成は、見かけ上、一つのトランザクションとして処理することにより、通常の処理によるデータ更新と矛盾を発生しない。

【００２４】

本第１の発明では、再編成を以上のような方法で実施するため、データに対するアクセスを停止することなく再編成が自動的に行えることになる。本発明では、再編成のために、一時点で１個または複数個のブロックに対するアクセスが排他待ちになってしまうが、それ以外のブロックに対しては常にアクセスが可能となる。また、再編成中の１個または複数個のブロックに対するアクセスが排他待ちになってしまう、という状態は、通常のレコード更新時にも発生することであり、特段の問題ではない。

【００２５】

本第１の発明において、再編成をしている最中に、レコードのアクセスを行うためには、再編成ポインターを使用する。この再編成ポインターは、現用のロケーション・テーブルと新規のロケーション・テーブルの双方に、１つずつ設ける。再編成ポインターは、ロケーション・テーブルとブロックの再編成が、どこまで進んでいるかを示すためのものである。

本第１の発明において、再編成をしている最中に、レコードを主キーで検索、格納、更新、削除する場合に、目的の主キー値が再編成ポインターが指しているエントリーのプライマリー・ブロック及びオーバーフロー・ブロックに含まれるレコードの主キー値より大きいか小さいかを比較し、目的の主キー値が、再編成ポインターが指しているブロックに格納されているレコードの主キー値以上のときには、現用のロケーション・テーブルを、前記主キー値未満のときには新規のロケーション・テーブルを使用して目的のレコードの検索を行う。

ここで、目的のレコードの検索を行う場合において、現用のロケーション・テーブルを使用するときは、現用のロケーション・テーブルの再編成ポインターとラスト・ポインターの間に対してバイナリー・サーチを行う。なお、ラスト・ポインターとは、ロケーション・テーブルを予め大きく確保しておき、その中で、どのエントリーまでが使用されているか指し示すためのものである。

【００２６】

一方、目的のレコードの検索を行う場合において、新規のロケーション・テーブルを使用するときは、新規ロケーション・テーブルの先頭ポインターと、新規ロケーション・テーブル用の再編成ポインターの間に対して、バイナリー・サー

チを行う。

本第１の発明では、このように再編成ポインターを使用することで、再編成中であっても目的のレコードを検索することが可能となる。

なお、レコードの更新、追加、挿入、削除の場合も、まず、目的のブロックを探す必要があることから、上述したのと同様のロジックで実現可能である。

【００２７】

（第２及び第３の発明の概要）

本第２の発明は、代替キー・テーブルの再編成に係るものである。

また、本第３の発明は、代替キー・テーブルに代替キー・ロケーション・テーブルを追加して、代替キー・テーブルと代替キー・ロケーション・テーブルの再編成に係るものである。本第２及び第３の発明では、代替キー・テーブルの再編成について、以下のようにしている。すなわち、本第２及び第３の発明では代替キー・テーブルの再編成に関しても、ロケーション・テーブル及びブロックの再編成に関する手段と同様な手段により、解決が可能である。

【００２８】

本発明者が提案したデータ格納検索方式では、代替キーと当該代替キー値のレコードが格納されているブロック番号と当該レコードの主キーとからなるエントリーを代替キーの順番に並べて複数格納可能とするとともに、予め同一の大きさで必要数を連続して確保できる代替キー・ブロックを使用し、代替キー・テーブルのエントリーは代替キー・ブロック中に代替キーの順番で並ぶように格納されており、同一の代替キーを持つ代替キー・テーブルのエントリーは同一の代替キー・ブロックに格納されており、同一の代替キーを持つエントリーの数が多いかもしくは代替キーの挿入により、代替キー・ブロックに格納できないときに、代替キー・オーバーフロー・ブロックを代替キー・ブロックに追加してエントリーを格納しており、初期のレコードの数が最終格納予定レコード数に比較して少ない場合は、代替キー・ブロックと同様な構造を持つプレ代替キー・ブロックを１段階以上使用できるようにしたものである。

【００２９】

そして、本発明者が提案したデータ格納検索方式では、代替キー・テーブルは

、上述のようにそれ自体が連続領域に格納されており、代替キー・テーブルに対して、バイナリー・サーチを行うことにより、目的の代替キー・ブロックを検索することとしている。これに対して、本第３の発明では、更に、代替キー・テーブルに対して、代替キー・ロケーション・テーブルを新たに追加することにより、更に再編成を効率化できる方法も考案している。

ここで、代替キー：テーブルは、代替キー・ブロックと代替キー・オーバーフロー・ブロックとから構成されている。

【００３０】

（第２の発明）

次に、本第２の発明では、現用代替キー・テーブルに対して、新規に代替キー・テーブルを作成し、現用代替キー・テーブルのブロックを現用代替キー・テーブルから現用代替キー・テーブルに順次、書き移していく。この順次書き移してゆく際に、代替キー・ブロックに連結されている代替キー・オーバーフロー・ブロックは、連結を切り離し、新たに代替キー・ブロックとして、現用代替キー・テーブルに書き移しを行う。

【００３１】

本第２の発明において、フラグメンテーションの解消は、複数の代替キー・ブロック及び代替キー・オーバーフロー・ブロックの格納率を調べて、複数の代替キー・ブロック及び代替キー・オーバーフロー・ブロックに対して、代替キー・ブロック及び代替キー・オーバーフロー・ブロック相互間でレコードの移動を行い、必要に応じて、新規に代替キー・ブロックを追加するか、使用されていた代替キー・ブロックまたは代替キー・オーバーフロー・ブロックを未使用にし、使用している代替キー・テーブルを順次、現用代替キー・テーブルから現用代替キー・テーブルに書き移すことによって実現している。

【００３２】

本第２の発明において、適正な初期格納率の確保は、オーバーフロー・ブロックの解消と、フラグメンテーションの解消を行う際に同時に実施し、ブロック内のエントリーが占める容量が、初期格納率となるように、エントリーの移動を行う。

上記の動作は、１個または複数個の代替キー・ブロックを対象にして、排他を掛けた状態で実施する。

このように再編成を行うと、一時点では、１個または複数個の代替キー・ブロックに対するアクセスが待ちになってしまうが、それ以外の代替キー・ブロックに対してはアクセスが可能となる。また、１個または複数個の代替キー・ブロックに対するアクセスが待ちになってしまう、という状態は、通常のレコード更新時にも発生することであり、特段の問題ではない。また、前述したように、再編成を一つのトランザクションとして処理することにより、矛盾を発生することも無い。

【００３３】

本第２の発明において、再編成をしている最中に、レコードのアクセスを行うには、再編成ポインターを使用する。この再編成ポインターは、現用代替キー・テーブルと現用代替キー・テーブルの双方に、１つずつ設ける。再編成ポインターは、代替キー・テーブルと代替キー・ブロックの再編成が、どこまで進んでいるかを示すためのものである。

本第２の発明において、レコードを、再編成を行っている代替キーで検索する場合に、目的の代替キー値が再編成ポインターが指しているエントリーの代替キー・ブロックに含まれるエントリーの代替キー値より大きいか小さいかを比較し、目的の代替キー値が、再編成ポインターが指している代替キー・ブロックに格納されているエントリーの代替キー値以上のときには、現用代替キー・テーブルを、前記エントリーの代替キー値未満のときには現用代替キー・テーブルを使用して目的のエントリーの検索を行う。

【００３４】

ここで、レコードを、再編成を行っている代替キーで検索する場合に、現用代替キー・テーブルを使用するときは、現用代替キー・テーブルの再編成ポインターとラスト・ポインターの間に対してバイナリー・サーチする。

一方、レコードを、再編成を行っている代替キーで検索する場合に、現用代替キー・テーブルを使用するときは、新規代替キー・テーブルの先頭ポインターと、現用代替キー・テーブル用の再編成ポインターの間に対して、バイナリー・サー

チを行う。

本第２の発明では、このようにすることで、目的のエントリーを検索することが可能となる。また、検索したレコードを、更新、削除が可能である。

【００３５】

（第３の発明）

本第３の発明は、代替キー・テーブルに対して、代替キー・ロケーション・テーブルを保持する形式に対して代替キー・テーブルの再編成を行うものであり、具体的には以下のようになる。

本第３の発明において、代替キー・ロケーション・テーブルのエントリーは、そのエントリーが指している代替キー・ブロックの番号とアドレスの他に、必要に応じて、そのエントリーが指している代替キー・ブロック及び、その代替キー・ブロックに連結されている代替キー・オーバーフロー・ブロックすべてに格納されているレコードの、主キー値の最小値と最大値の双方、または、何れか一方を保持している。

ここで、本第３の発明では、現用代替キー・ロケーション・テーブルに対して、新規に代替キー・ロケーション・テーブルを作成し、現用代替キー・ロケーション・テーブルのエントリーを新規の代替キー・ロケーション・テーブルに順次、書き移していく。本第３の発明では、この順次書き移してゆく際に、代替キー・オーバーフロー・ブロックの連結を切り離し、新規の代替キー・ロケーション・テーブルに新しいエントリーを追加し、新規の代替キー・ロケーション・テーブルの代替キー・ブロックとする。

【００３６】

本第３の発明では、フラグメンテーションの解消を次のようにしている。すなわち、フラグメンテーションの解消は、複数の代替キー・ブロック及び代替キー・オーバーフロー・ブロックの格納率を調べて、複数の代替キー・ブロック及び代替キー・オーバーフロー・ブロックに対して、代替キー・ブロック及び代替キー・オーバーフロー・ブロック相互間でレコードの移動を行い、必要に応じて、新規に代替キー・ブロックを追加するか、使用されていた代替キー・ブロックまたは代替キー・オーバーフロー・ブロックを未使用にし、代替キー・ロケーショ

ン・テーブルのエントリーから削除することにより実現している。

【００３７】

適正な初期格納率の確保は、オーバーフロー・ブロックの解消と、フラグメンテーションの解消を行う際に同時に実施し、ブロック内のエントリーが占める容量が、初期格納率となるように、エントリーの移動を行う。

上記の動作は、１個または複数個の代替キー・ブロックを対象にして、排他を掛けた状態で実施する。

このように再編成を行うと、一時点では、１個または複数個の代替キー・ブロックに対するアクセスが待ちになってしまうが、それ以外の代替キー・ブロックに対してはアクセスが可能となる。また、数個から複数個の代替キー・ブロックに対するアクセスが待ちになってしまう、という状態は、通常のレコード更新時にも発生することであり、特段の問題ではない。

【００３８】

本第３の発明において、再編成をしている最中に、レコードのアクセスを行うには、再編成ポインターを使用する。この再編成ポインターは、現用代替キー・ロケーション・テーブルと新規の代替キー・ロケーション・テーブルの双方に、１つずつ設ける。再編成ポインターは、代替キー・ロケーション・テーブルと代替キー・ブロックの再編成が、どこまで進んでいるかを示すためのものである。

本第３の発明において、再編成をしている最中に、レコードを、再編成を行っている代替キーで検索する場合に、目的の代替キー値が再編成ポインターが指しているエントリーの代替キー・ブロックに含まれるエントリーの代替キー値より大きいか小さいかを比較し、目的の代替キー値が、再編成ポインターが指している代替キー・ブロックに格納されているエントリーの代替キー値以上のときには、現用代替キー・ロケーション・テーブルを、前記エントリーの代替キー値未満のときには新規の代替キー・ロケーション・テーブルを使用して目的のエントリーの検索を行う。

ここで、再編成を行っている代替キーに対して、レコードの検索を行う場合、現用代替キー・ロケーション・テーブルを使用するときは、現用代替キー・ロケーション・テーブルの再編成ポインターとラスト・ポインターの間に対してバイ

ナリー・サーチする。

【００３９】

一方、再編成を行っている代替キーに対して、レコードの検索を行う場合、新規の代替キー・ロケーション・テーブルを使用するときには、新規の代替キー・ロケーション・テーブルの先頭ポインターと、新規の代替キー・ロケーション・テーブル用の再編成ポインターの間に対して、バイナリー・サーチを行う。

本第３の発明では、このようにすることで、目的のエントリーを検索することが可能となる。なお、エントリーの更新、追加、挿入、削除も、まず、目的の代替キー・ブロックを探す必要があり、同様のロジックで実現可能である。

【００４０】

【発明の実施の形態】

以下、本第１の発明ないし第３の発明の実施の形態について図面を参照して説明をするが、その前に本発明の第1の発明ないし第3の発明の基礎となった事項についてまず説明する。

【００４１】

＜本第１の発明ないし第３の発明の基礎＞

本第１の発明ないし第３の発明は、特開平１１－０３１０９６号公報及び米国特許第６４１５３７５号公報に記載されている発明の発想を踏まえて、基本部分をそのまま利用しながら、データ格納検索方式の運用を止めることなく、自動的に再編成を実施することを目的とする。

【００４２】

また、同様に本発明者が発明した特開２００１-３５６９４５号公報に記載されている「データ・バックアッ プ・リカバリー方式」では、プライマリー・システムに対してセカンダリー・システ ムを１つ以上用意し、ロケーション・テーブル、ブロック、代替キー・テーブルを１組としてセカンダリー・システムを構成するか、ブロックのみを保持してバックアップし、リカバリー時にそのバックアップを使用する方式を示している。この「データ・バックアップ・リカバリー方式」を使用した場合にも、本第１の発明ないし第３の発明は適用可能であることも説明する。

【００４３】

以下に、特開平１１－０３１０９６号公報に記載されたデータ格納検索方式に関して説明を行う。特開平１１－０３１０９６号公報に記載された「データ格納検索方式」の特徴の第１は、ロケーション・テーブルという、レコードを格納するためのブロック（プライマリー・ブロック及びオーバーフロー・ブロック）を、ロケーション・テーブルというフラットな（階層構造を持たない）テーブルを用いて管理している点である。ブロック間では、あるブロックより前にあるブロックに格納されているレコードの主キー値は、あるブロックに格納されているレコードの主キー値より小さい。ブロックには、レコードをそのレコードの主キーの順番に格納するようにする。これは、プライマリー・ブロック内でも、プライマリー・ブロックに連結されているオーバーフロー・ブロック内でも、また、その相互間でも同様である。このようにすることで、ブロック内のレコードを検索する際の効率が向上する。

【００４４】

「主キー」とは、一つの種類のデータ上でユニークなキーであり、レコードに一つ必要となるものである。例えば、従業員マスターにおける、従業員コード、取引先マスターにおける取引先コードなどである。「代替キー」とは、データ上でノン・ユニークなキーであり、レコード中に複数種類の代替キーが存在してもよい。例えば、従業員マスターにおける、従業員氏名や所属、入社年月日などである。

ブロック内にレコードの挿入がある場合、そのブロックに格納できなくなった場合には、そのブロックに対して、オーバーフロー・ブロックを追加し、双方を連続して使用することにより、レコードの格納を行う。オーバーフロー・ブロックが一つで足りなくなった場合には、そのオーバーフロー・ブロックに対して更にオーバーフロー・ブロックを連結し、順次オーバーフロー・ブロックを連結することにより、レコード挿入が幾つでも可能としてある。

【００４５】

ロケーション・テーブルはプライマリー・ブロックの管理のみを行い、オーバーフロー・ブロックはプライマリー・ブロックの従属として、ロケーション・テ

ーブルでは管理されない。このため、オーバーフロー・ブロックが発生しても、ロケーション・テーブルの構造的な変更は発生しない。

上記「データ格納検索方式」の特徴の第２は、主キーによる検索には、ロケーション・テーブルをバイナリー・サーチして目的のブロックを求めるという方法により、旧来のインデックスを使用せず、インデックス管理を効率化し、検索・格納を高速化した点である。

【００４６】

上記「データ格納検索方式」の特徴の第３は、代替キー用に代替キー・テーブルという、これもフラットなテーブルを採用する。「代替キー」とは、既に説明したように、データ上でノン・ユニークなキーであり、レコード中に複数種類の代替キーが存在してもよい。例えば、従業員マスターにおける、従業員氏名や所属、入社年月日などである。

【００４７】

代替キーの追加や変更によりキー値のエントリーが増加しても、代替キー・テーブル・ブロックに代替キー・オーバーフロー・ブロックを追加することで、旧来のインデックスにあったような分割をなくし、代替キー用のインデックス管理を効率化したことである。また、代替キー・オーバーフロー・ブロックが一つで不足する場合には、その代替キー・オーバーフロー・ブロックに対して、更に、代替キー・オーバーフロー・ブロックを連結することにより、代替キー・エントリーの挿入に関する制限を無くしている。

上記「データ格納検索方式」の特徴の第４は、代替キーによる検索について、代替キー・テーブルをバイナリー・サーチにより検索することにより高速化したものである。

更に、上記「データ格納検索方式」の特徴の第５は、代替キー・テーブルに対するキー値の追加変更が多く行われると、代替キー・オーバーフロー・ブロックが増加するため、検索効率が低下する可能性があるが、プレ代替キー・テーブルを使用することにより、検索効率が低下しないような工夫がされている。

【００４８】

本第１の発明ないし第３の発明では、特開平１１－０３１０９６号公報の「デ

ータ格納検索方式」で示された、データ格納用ファイル（ブロックの集合）に対して、ロケーション・テーブル、及び、代替キー・テーブルを用いてレコードの検索を行うことになっている。

また、ブロックに対してレコードの挿入や追加に伴い、オーバーフロー・ブロックが追加されることや、レコードの削除に伴いブロックに空き領域が発生する。このように、アクセス効率とブロックの格納効率が低下する問題を、再編成により、レコードのアクセス効率、及び、格納効率を最大限にすることを目的としている。

更に、代替キー・テーブルにおいては、同様に、レコードの追加・更新・削除に伴って、代替キー値が変更される場合もあり、代替キー・オーバーフロー・ブロックが追加されることや代替キー・ブロック及び、代替キー・オーバーフロー・ブロックに空き領域が発生する。

これによって、アクセス効率と代替キー・エントリーの格納効率が低下する問題を、再編成により、エントリーのアクセス効率、及び、格納効率を最大限にすることを目的としている。

【００４９】

本第１の発明ないし第３の発明では、この「データ検索格納方式」で定義されているロケーション・テーブル、ブロック、代替キー・テーブル（種類ごとに１つ）の組合せ（１セット）をプライマリー・システムと呼ぶ。

実際の適用においては、プライマリー・システムは、複数のセットから構成されると考えられるが、１セットに対して再編成する方式を、他のセットに適用すればよい。

本第１の発明ないし第３の発明では、所謂、データベースを念頭におくと理解しやすいが、 データベースのみを対象とするものではなく、データ格納検索方式及びシステム全般を対象としている。従来のコンピューターでは、外部記憶装置に格納されたプログラムやデータを、メイン・メモリーにロードして実行するという方式を採っていた。このため、外部のデータベースと、内部のメイン・メモリーは画然と分かれていたが、今後、不揮発性メモリーが普及すると、外部記憶装置にも高速なランダム・アクセス・メモリーが採用されることになろう。そ

うなった場合に、外部記憶と内部記憶を分けなければならない理由が無くなる。データが、外部記憶装置以外のどこに格納されていても、本方式及びシステムの適用が可能である。

【００５０】

[再編成の目的と理由]

再編成の目的は、次の３項目である。(i)　オーバーフロー・ブロックの解消、(ii)フラグメン テーションの解消と、(iii)適正な初期格納率の確保である。

以下、再編成の目的と理由につい て上記３項目について説明する。

【００５１】

[再編成の目的（その１）：(i)オーバーフロー・ブロックの解消]

「オーバーフロー・ブロック」は、レコードの挿入によって発生するものである。あるプライマリー・ブロックにレコードが満杯に入っており、更に、そのプライマリー・ブロックに対して、レコードを挿入しようとする場合に、そのままでは挿入できないことになってしまう。このような挿入を可能とするために、当該プライマリー・ブロックに対して、オーバーフロー・ブロックを割り当て、必要な数のレコードを、プライマリー・ブロックからオーバーフロー・ブロックに移動し、元のプライマリー・ブロックに目的のレコードを挿入することにより、レコードを格納することが可能となる。このようなオーバーフロー・ブロックの使用に関しては、「特開平１１－３１０９６号公報に記載されている「データ格納検索方式」に述べられている。

【００５２】

ところが、オーバーフロー・ブロックがプライマリー・ブロックに連結されていると、ロケーション・テーブルをバイナリー・サーチで検索して、目的のプライマリー・ブロックを探した後に、当該プライマリー・ブロックとそれに連結されているオーバーフロー・ブロック内で目的のレコードを探すための動作が、プライマリー・ブロックのみが存在する場合に比較して、余分に必要となる。オーバーフロー・ブロックをプライマリー・ブロックとして、ロケーション・テーブルで管理するようにすることが、早いアクセスを実現する上で、必要なことになる。

また、レコードをブロック内で主キーの順番に並べるために、レコードの挿入ではレコードの移動が必要になるが、オーバーフロー・ブロックが多いと、移動対象レコードの数が増加し、効率上の問題になるものであった。

【００５３】

[再編成の目的（その２）：(ii)フラグメンテーションの解消]

次は、フラグメンテーションの解消である。「フラグメンテーション」とは、格納領域の分散化である。ブロック（プライマリー・ブロック及びオーバーフロー・ブロック）に格納されているレコードが不要になって、消去するとその分、ブロックの格納領域が空くことになる。挿入されるレコードが無い場合には、格納領域が空いたまま無駄になってしまう。このような無駄を無くすために、次のブロックに格納されているレコードを前のブロックに移動して、ブロック内にレコードが適正な数だけ、格納されるようにすることにより、格納領域を使用する上での無駄が解消できる。

【００５４】

また、オーバーフロー・ブロックが発生して、当該オーバーフロー・ブロックの格納容量に比較して、少数のレコードが格納された状態で、レコード挿入がその後発生しないと、その空き容量も、使用されない無駄な格納領域になってしまう。

更には、ブロック単位で考えると、使用していたブロックが使用されなくなることにより、ブロックが格納領域上で散在した状態になって再使用されなくなると、格納領域全体から見てのフラグメンテーションと言える。本発明の基礎では、未使用ブロックの再利用に関しては、ブロック内のフラグメンテーションの解消とは解決案を分けて説明している。

【００５５】

[再編成の目的（その３）：(iii)適正な初期格納率の確保]

次に適正な初期格納率の確保がある。「初期格納率」とは、最初にブロックを作成してレコードを書き込む場合に、予め一定割合のブロック内領域を空けておき、オーバーフロー・ブロックの発生を防止するためのものである。

初期に、ブロックにレコードを格納する場合に、ブロックの容量の１００％に

対してレコードを格納することは可能であるが、そのように満杯になるまでレコードを格納した状態で、レコードが挿入されると、直ちにオーバーフロー・ブロックが発生することになる。オーバーフロー・ブロックは、検索効率の低下に繋がるのみならず、オーバーフロー・ブロックに格納されるレコードが少ない場合には、オーバーフロー・ブロックの空き領域が多くなってしまい、前述のフラグメンテーションの原因となる。このような状態になることを避けるために、例えば、当初、プライマリー・ブロックにレコードを格納する場合に、格納容量の９０％までを限度として、レコードを格納しておき、その後、挿入レコードが発生した場合に、その空領域を使用してレコードの格納を行うことにより、直ちにオーバーフロー・ブロックを発生させないようにしておくことが可能である。

このように、予め一定割合のブロック内領域を挿入レコードのために空けておくというのが適正な初期格納率の考えかたであるが、これは従来から良く知られた技法の一つである。

【００５６】

[再編成の目的（その３の詳細説明）：適正な初期格納率の確保の詳細]

適正な初期格納率を適用するのは、（ａ）データベースを最初に作成する場合、または、（ｂ）ブロックを新たに確保してそのブロックに追加型でレコードを格納する場合、（ｃ）既に確保されているブロックの格納率が適正な格納率を下回っているときに、当該ブロックにレコードを挿入する場合、または、（ｄ）再編成を行う場合、の４つである。

【００５７】

上記（ｂ）のブロックを新たに確保してそのブロックに追加型でレコードを格納する場合、または、上記（ｃ）の既に確保されているブロックの格納率が適正な格納率を下回っている場合に、当該ブロックにレコードを挿入する場合の２つは、厳密な定義としては、別個のものであるが、実用上は同じロジックで実現できる。

上記（ｂ）のブロックを新たに確保してブロックにレコードを格納する場合を、もう少し詳しく説明すると、新たにブロックを確保するのは、ロケーション・テーブルのラスト・ポインターが指している直前のエントリー（最終エントリー

）が指しているブロックが適正な格納率を超えており、そのブロックに格納されているレコードより、大きな主キー値を持っているレコードが格納対象となった場合に、新たにプライマリー・ブロックを割り当てて、新たに割り当てたプライマリー・ブロックに、当該レコードを格納する。

【００５８】

この新たなプライマリー・ブロックに対して、追加型でレコードが書き込まれる場合は、適正な初期格納率まで、レコードの格納を行い、適正な格納率を超えたら、更に、次のプライマリー・ブロックを割り当てる。このように、プライマリー・ブロックの割り当て後は、適正な初期格納率に達するまで、レコードの格納を行う。これは、当該ブロックに対して、挿入型のレコード格納であっても、同様であり、適正な格納率に達するまでは、当該プライマリー・ブロックに対して格納を行う。

【００５９】

このように新たにプライマリー・ブロックを割り当てた場合に、ラスト・ポインターを順次移動するのは、当然のことである。

また、新たにブロックを割り当てるのは、上記以外に、オーバーフロー・ブロックの割り当てがある。オーバーフロー・ブロックに対してレコードを格納する場合も、プライマリー・ブロックの場合と同様で、適正な初期格納率に達するまで、レコードの格納を行う。適正な初期格納率に達した場合に、次のオーバーフロー・ブロックを新たに割り当てて、その新しいオーバーフロー・ブロックにレコードの格納を行う。

【００６０】

既に確保されているブロックの格納率が適正な格納率を下回っている場合に、当該ブロックにレコードを挿入する場合というのは、以下に述べるようなことである。

あるブロック（プライマリー・ブロックであっても、オーバーフロー・ブロックであってもよい）にレコードが格納されていた状態に対して、レコードの消去が発生すると、ブロック内から消されて、その分の格納領域が空くことになる。この空領域が、適正な初期格納率を下回っていると、その空領域は無駄に使用さ

れている、と考えられる。このようなブロックに対して、レコードが挿入される場合には、適正な初期格納率に達するまで、レコードを格納することが、格納領域の有効利用上、有用である。

【００６１】

このような、適正な初期格納率の考え方は、あくまでも、次のブロックを新たに割り当てるか否かの判断に使用するためのもので、あるプライマリー・ブロックにオーバーフロー・ブロックが割り当てられていて、そのプライマリー・ブロックに対して、レコードの挿入が発生した場合には、格納率が１００％に達するまで格納を行う。

以上では、ロケーション・テーブルとブロック、オーバーフロー・ブロックに対する考え方を中心に記述したが、代替キー・ブロック、代替キー・オーバーフロー・ブロックに対しても同様の問題を抱えており、後述するように、代替キー・ブロックの再編成も本発明の対象である。

【００６２】

[統計手法の利用による再編成の自動化]

自動再編成を実施するには、以下のように統計手法を用いるのが有効である。このような統計手法は旧来の方式でも用いられてきた方法である。これは、ソフトウエアによって自動的に実行することが可能である。

【００６３】

従来の方式であるデータ格納システムに対して、統計手段を追加する。この統計手段には、ブロックに対するオーバーフロー・ブロックの発生状況を把握するために、次の(a-i)～(a-v)に示す項目を調査する。なお、必ずしも(a-i)～(a-v)に示す項目の総てを調査する必要はなく、目的によって選択すればよい。

(a-ｉ) 全体のブロック数。

(a-ii) プライマリー・ブロック数。

(a-iii)オーバーフロー・ブロック数。

(a-iv) 一つのプライマリー・ブロックあたりのオーバーフロー・ブロック数の最大。

(a-v) 一つのプライマリー・ブロックあたりのオーバーフロー・ブロック

数の標準偏差。

【００６４】

更に、フラグメンテーションと適正な初期格納率に関する次の(b-i)及び(b-ii)で示すデータ も有効であるが、適正な初期格納率は、前述したように、ブロックに対する初期状態に関するもので、フラグメンテーションが、より重要である。ブロックはプライマリー・ブロックとオーバーフロー・ブロックの双方を含む。

(b-ｉ) ブロックの総格納可能領域の大きさ。

(b-ii) ブロックの実際に使用されている格納領域の大きさ。

【００６５】

また同様に、代替キー・テーブルの代替キー・ブロックに対する代替キー・オーバーフロー・ ブロックの発生状況を把握するために、次の(c-i)～(c-iv)に示す項目を調査する。必ずしも総ての項目を調査する必要はなく、目的によって選択すればよい。

(c-i)　代替キー・ブロック数。

(c-ii) 代替キー・オーバーフロー・ブロック数。

(c-iii)一つの代替キー・ブロックあたりの代替キー・オーバーフロー・ブロック数の最大。

(c-iv) 一つの代替キー・ブロックあたりの代替キー・オーバーフロー・ブロック数の標準偏差。

【００６６】

同様に、フラグメンテーションと適正な初期格納率に関する次の(d-i)及び(d-ii)に示すデータも有効であるが、適正な初期格納率は、前述したように、代替キー・ブロックに対する初期状態に関するもので、フラグメンテーションに関する情報が、より重要である。

(d-i)代替キー・ブロックと代替キー・オーバーフロー・ブロックの双方の総格納可能領域の大きさ。

(d-ii)代替キー・ブロックと代替キー・オーバーフロー・ブロックの双方の実際に使用されている格納領域の大きさ。

【００６７】

また、ロケーション・テーブル、各々の代替キー・テーブルに対するアクセス数を数えることが好ましい。これは、オーバーフローが発生していても、アクセスが少ない場合には、再編成を行わない、という選択を可能とするためである。

上記の項目の数字を調査し、予め定めた閾値に達したら自動的に再構成を行う。

これから述べるように、本第１の発明ないし第３の発明に係るデータ及びデータベースの無停止自動再編成システムでは、プライマリー・システムの運用を止めずに、運転しながら再構成を行うことが可能である。また、上述した「データ・バックアップ・リカバリー方式」を使用している場合にも、プライ マリー・システムと セカンダリー・システムとの間に矛盾を発生することなく、再編成が可能である。

【００６８】

この自動再編成システムを使用する場合には、再構成がシステムの稼動負荷が高い時間帯に行われると、システム全体のパフォーマンスが低下するため、稼動負荷が低い時に行うことが好ましいが、この自動再編成システムは、後述するように、いつでも再編成を中断して、再開することが可能である。これにより、旧来の方法のように、長時間の運用停止を行う必要がなく、システム負荷の低い状態の時に、逐次、進めることが可能である。

【００６９】

［オーバーフロー・ブロックの形式の選択］

（オーバーフロー・ブロックの形式に関して）

上述した「データ格納検索方式」では、オーバーフロー・ブロックの形式に関して、３種類の提案を行っている。すなわち、第１にプライマリー・ブロックと同じ大きさのブロックとする、第２にプリマリー・ブロックと異なった大きさのブロックとする、第３に複数のプライマリー・ブロックから管理されるようにする、の３種である。

本発明では、プライマリー・ブロックと同一の大きさのオーバーフロー・ブロックを使用することを前提とする。何故ならば、オーバーフロー・ブロックの大

きさが、プリマリー・ブロックと異なった大きさのブロックとする場合には、再編成によって、ブロックの大きさを変更する必要があること、プライマリー・ブロックと異なる大きさのオーバーフロー・ブロックが未使用ブロックになった場合には、再使用に関して制限が大きくなることから好ましくない。また、一つのオーバーフロー・ブロックが複数のプライマリー・ブロックから管理されるようにする場合には、再編の負荷が増大することから、同様に好ましくない。

【００７０】

[再編成時の必要領域]

（２つのロケーション・テーブルを保持）

この自動再編成システムによる自動再編成を実施する場合には、一時点で、再編成対象のロケーション・テーブル、代替キー・テーブルを２つ持つ必要があるため、領域の余裕が必要である。

代替キー・テーブルが３種類（Ａ，Ｂ，Ｃ）ある場合（図１参照）を例にとって説明する。

ロケーション・テーブルＬＣと代替キー・テーブルＡ、Ｂ、Ｃの何れについて、再編成をする必要があるかの判断を行う。これは、前述した、オーバーフロー・ブロックの発生数、標準偏差等を調べて、一定の閾値を超えているものを対象にする。また、ロケーション・テーブルの他に、同時に、代替キー・テーブルが一つ以上、再編成対象になった場合には、すべての再編成を同時に行うのではなく、まず、ロケーション・テーブルの再編成を行い、次に、代替キー・テーブルの再編成を、種類毎に順次実施する。Ａ，Ｂ，Ｃの３種類が同時に再編成対象になった場合には、まず、Ａの再編成を行い、完了したらＢの再編成を行い、Ｂの再編成が完了したらＣの再編成を行う。

つまり、新旧２つのテーブル用の領域が必要であるが、ロケーション・テーブルと代替キー・テーブルのすべてに対して同時に必要なのではなく、一時点では、再編成の対象となるテーブルに対して、新旧の領域が必要となるので、領域が多大に必要になるわけではない。

【００７１】

＜本第１の発明の実施の形態＞

図１は、本第１の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムが適用されるプライマリー・システムの一例を示す構成図である。

この図１において、前記プライマリー・システム１は、当該ロケーション・テーブルＬＣと、ブロック１０と、代替キー・テーブル１１Ａと、代替キー・テーブル１１Ｂと、代替キー・テーブル１１Ｃから構成されている。なお、図示しないが同様の構成のセカンダリー・システムも存在するが、説明の便宜上説明を省略する。

前記ロケーション・テーブルＬＣは、図１に示すように、当該ロケーション・テーブルＬＣに記述されているブロック番号に該当するブロック１０の位置を指し示している。

【００７２】

図２は、図１で示したプライマリー・システムの内、本第１の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムが適用される部分のみを示した概略図である。

この図２では、図１のプライマリー・システム１について、ブロック１０をプライマリー・ブロック１２とオーバーフロー・ブロック１３とオーバーフロー・ブロック１４とに分け、これらとロケーション・テーブルＬＣとで構成したことを示すために、プライマリー・システム１として説明することにする。すなわち、プライマリー・システム１は、ロケーション・テーブルＬＣと、プライマリー・ブロック１２と、オーバーフロー・ブロック１３と、オーバーフロー・ブロック１４とから構成されている。

【００７３】

ロケーション・テーブルＬＣの１番目は、プライマリー・ブロック１２のブロック番号「０」を指し示している。

ロケーション・テーブルＬＣの２番目はプライマリー・ブロック１２のブロック番号「１」を指し示しており、このプライマリー・ブロック１２はオーバーフロー・ブロック１３のブロック番号「１－２」を、オーバーフロー・ブロック１３はオーバーフロー・ブロック１４のブロック番号「１－３」を、それぞれ指し

示している。

ロケーション・テーブルＬＣの３番目は、プライマリー・ブロック１２のブロック番号「２」を指し示している。

ロケーション・テーブルＬＣの４番目は、プライマリー・ブロック１２のブロック番号「３」を指し示している。

ロケーション・テーブルＬＣの５番目は、プライマリー・ブロック１２のブロック番号「４」を指し示している。

ロケーション・テーブルＬＣの６番目はプライマリー・ブロック１２のブロック番号「５」を指し示しており、このプライマリー・ブロック１３はオーバーフロー・ブロック１３のブロック番号「５－２」を指し示している。

ロケーション・テーブルＬＣの７番目はプライマリー・ブロック１２のブロック番号「６」を指し示しており、このプライマリー・ブロック１２はオーバーフロー・ブロック１３のブロック番号「６－２」を、オーバーフロー・ブロック１３はオーバーフロー・ブロック１４のブロック番号「６－３」を、それぞれ指し示している。

ロケーション・テーブルＬＣの８番目は、プライマリー・ブロック１２のブロック番号「７」を指し示している。

ロケーション・テーブルＬＣの９番目は、プライマリー・ブロック１２のブロック番号「８」を指し示している。

【００７４】

以下、上記ロケーション・テーブルＬＣは、プライマリー・ブロック１２をそれぞれ指し示すことになる。

図２では、上記プライマリー・システム１は、プライマリー・ブロック１２のブロック番号「１」、ブロック番号「２」、ブロック番号「５」、ブロック番号「６」にオーバーフローが生じている。

このようなプライマリー・システム１のロケーション・テーブルＬＣ及びプライマリー・ブロック１２、オーバーフロー・ブロック１３、オーバーフロー・ブロック１４について再編成を次のように行う。

【００７５】

次に、本第１の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムの動作について、図２を基に図３を参照して説明する。

ここに、図３は、本第１の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムの動作を説明するための図である。

【００７６】

［ロケーション・テーブル及びブロックの再編成：オーバーフローの解消］

まず、ロケーション・テーブルＬＣが再編成の対象になった場合を説明する。

この図２及び図３において、「ロケーション・テーブルＬＣの再編成」とは、次に述べるようなことである。

ロケーション・テーブルＬＣが管理しているブロック１０は、プライマリー・ブロック１２のみである。オーバーフロー・ブロック１３はプライマリー・ブロック１２に管理されている。言い換えれば、オーバーフロー・ブロック１３のアドレスをプライマリー・ブロック１２が保持している、ということである。このため、ロケーション・テーブルＬＣを利用して主キーで検索を行う場合に、対象となるプライマリー・ブロック１２はロケーション・テーブルＬＣに対してバイナリー・サーチを行うことにより該当のブロック１５が見つかった場合、該当のブロック１５内では、対象となるレコードを探す必要がある。該当のプライマリー・ブロック１２に対してオーバーフロー・ブロック１３，１４が多数連結されていると、対象となるレコードを探すための時間が、プライマリー・ブロック１２（図１では、例えばブロック番号「０」、ブロック番号「３」、ブロック番号「４」、ブロック番号「７」、ブロック番号「８」）しか存在しない場合に比較して、オーバーフロー・ブロック１３（１４）の数に関連して、余分に掛かることになる。

【００７７】

これを避けるために、本第１の発明の実施の形態では、オーバーフロー・ブロックを無くして、すべてのオーバーフロー・ブロック１３，１４をプライマリー・ブロック（１２）にし、ロケーション・テーブル（ＬＣまたはＬＮ）で管理することにより、対象レコードを探す時間を、短縮できるようにしたものである。

また、レコードをブロック（１０）内で主キーの順番に並べるために、レコー

ドの挿入ではレコードの移動が必要になるが、オーバーフロー・ブロック（１３，１４）が多いと、移動対象レコードの数が増加し、効率上の問題になるものであった。

【００７８】

＜現用と新規の２つのロケーション・テーブルを使用して再編成を行う。＞

本第１の発明では、図３に示すように、現用ロケーション・テーブルＬＣと、新規のロケーション・テーブルＬＮとの２つを使用して再編成を行うようにしたものである。

本第１の発明において、オーバーフロー・ブロック（１３，１４，…）が幾つ発生しているかは、数えてあるので、ロケーション・テーブルＬＣのエントリー数と、オーバーフロー・ブロック（１３，１４，…）の数を足したものが、新しいロケーション・テーブルＬＮのエントリー数となる。ここで、本明細書では、再編成時点で使用しているロケーション・テーブルを以下「ＬＣ」と言い、新規のロケーション・テーブルを以下「ＬＮ」と言うことにする。エントリーは、再編成中にレコードの挿入によって増加する可能性があり、再編成後もデータ追加によって増加するので、必要数より多めに確保することが好ましい。

しかしながら、後述するように、他方で、フラグメンテーションを解消することにより、必要ブロック数が少なくなる場合もあるし、適正な初期格納率の確保によっても、必要ブロック数は変動するので、格納されているレコードの容量と適正な初期格納率を元に計算するのが、最も好ましい方法である。

【００７９】

本第１の発明の実施の形態では、第１の手段により、新規のロケーション・テーブルＬＮの容量を満たす連続領域を、プライマリー・システム１に確保する。

新規のロケーション・テーブルＬＮの領域の確保ができたら、図３に示すように、現用のロケーション・テーブルＬＣから新規のロケーション・テーブルＬＮに対して、そのエントリーを順次書き移していくのであるが、以下のような手順で行う。

【００８０】

＜再編成ポインター＞

本第１の発明の実施の形態では、第２の手段により次のような操作が行われる。まず、再編成ポインターと言うものを作成する。この再編成ポインターは、ロケーション・テーブル（ＬＣ，ＬＮ）の、どのエントリーまで再編成が終了したかを指し示すもので、現用のロケーション・テーブルＬＣ用と、新規のロケーション・テーブルＬＮ用の２つを用意する。この現用のロケーション・テーブルＬＣ用には再編成ポインターＲＰＬＣとし、新規のロケーション・テーブルＬＮ用には再編成ポインターＲＰＬＮとする。

ここで、現用のロケーション・テーブルＬＣ用の再編成ポインターＲＰＬＣの初期値は、現用のロケーション・テーブルＬＣの先頭番地、新規のロケーション・テーブルＲＰＬＮの初期値は新規のロケーション・テーブルＬＮの先頭番地とする。

また、上述したようにプライマリー・システム１のロケーション・テーブルＬＣの１番目，２番目，３番目，…は、図３に示すように、プライマリー・ブロック１２の所定ブロック番号を指し示している。

この図３において、まず、再編成ポインターＲＰＬＣの１番目のエントリーと、そのエントリーで管理しているプライマリー・ブロック１２（のブロック番号「０」）及びオーバーフロー・ブロックを排他する。この場合はオーバーフロー・ブロックが存在しないためプライマリー・ブロック１２（のブロック番号「０」）のみとなる。

次に、１番目のエントリー（ブロック番号「０」）を現用のロケーション・テーブルＬＣから新規のロケーション・テーブルＬＮに書き移す（図３のＳ１）。その際に、１番目のエントリー（ブロック番号「０」）で管理しているブロック（１２のブロック番号「０」）に、オーバーフロー・ブロックが連結されているか否かを確認する。連結されていない場合は、（現用の）再編成ポインターＲＰＬＣ（新規用の）再編成ポインターＲＰＬＮのアドレスを２番目のエントリーの先頭を指し示すように変更する。

【００８１】

図３ではブロック番号「０」のプライマリー・ブロック１２はオーバーフロー・ブロックが連結されていないので、（現用の）再編成ポインターＲＰＬＣ（新

規用の）再編成ポインターＲＰＬＮとも２番目のエントリーを指し示すように変更する。

（現用の）再編成ポインターＲＰＬＣの１番目のエントリーと、そのエントリーで管理しているプライマリー・ブロック（１２のブロック番号「0」）及びオーバーフロー・ブロックの排他を解除する。この場合はオーバーフロー・ブロックが存在しないためプライマリー・ブロックのみとなる。

次に、２番目のエントリー（ブロック番号「１」）の処理を行う。現用のロケーション・テーブルＬＣの２番目のエントリーと、そのエントリーで管理しているプライマリー・ブロック１２（ブロック番号「１」）及びオーバーフロー・ブロック１３（ブロック番号「１－２」）、オーバーフロー・ブロック１４（ブロック番号「１－３」）を排他する。現用のロケーション・テーブルＬＣの２番目のエントリーで管理されているプライマリー・ブロック１２には、オーバーフロー・ブロック１３，１４の２つが連結されている。このようにオーバーフロー・ブロック１３，１４がプライマリー・ブロック１２に連結されている場合には、次の様にする。現用のロケーション・テーブルＬＣの２番目のエントリーを新規のロケーション・テーブルＬＮの２番目のエントリーに書き移す（図３のＳ２）。この場合、単純に書き移すのではなく、ブロックに格納されているレコードの主キー値の最小値と最大値に関する変更を行う。更に、ロケーション・テーブルＬＣ及びＬＮのエントリーに、ブロック（１２，１３，１４）の主キー値の最大値、最小値を持っている場合、単に書き移すだけでは、ロケーション・テーブルＬＮのエントリーの主キー値の最小値と最大値が、ブロック（１２，１３，）に格納されているレコードの、主キー値の最小値と最大値と合致しなくなってしまうため、これを回避するためである。

【００８２】

仮に、現用のロケーション・テーブルＬＣの２番目のエントリーの主キー値の最小値が「００００」、最大値が「０２９９」だとする。そして、プライマリー・ブロック１２に格納されているレコードの主キー値の最小値が「００００」、最大値が「００９９」、１番目のオーバーフロー・ブロック１３に格納されているレコードの主キー値の最小値が「０１００」、最大値が「０１９９」、２番目

のオーバーフロー・ブロック１４に格納されているレコードの主キー値の最小値が「０２００」、最大値が「０２９９」とする場合に、次のようになる。

【００８３】

新規のロケーション・テーブルＬＮの２番目のエントリーの主キー値は、最小が「００００」、最大が「００９９」となる。ブロック１０のアドレスは、プライマリー・ブロック１２（ブロック番号「1」）のアドレスとするので、現用のロケーション・テーブルＬＮのアドレスと同じ値が入る（図３のＳ３）。次に、新規のロケーション・テーブルＬＮの３番目のエントリーには、１番目のオーバーフロー・ブロック１３のアドレス（ブロック番号「１－２」）と、主キー値の最小値には「０１００」、最大値には「０１９９」を入れる（図３のＳ３）。

新規のロケーション・テーブルＬＮの４番目のエントリーには、２番目のオーバーフロー・ブロック１４のアドレス（ブロック番号「１－３」）を入れ、主キー値の最小値は「０２００」、最大値は「０２９９」とする（図３のＳ４）。

【００８４】

次に、プライマリー・ブロック１２（ブロック番号「1」）中のオーバーフロー・ブロック・アドレスをリセットして、オーバーフロー・ブロック１３（ブロック番号「１－２」）がプライマリー・ブロック１２（ブロック番号「1」）に連結されていない状態にする（図３のＳ５）。次に１番目のオーバーフロー・ブロック１４中のオーバーフロー・ブロック・アドレスをリセットして、２番目のオーバーフロー・ブロック１４（ブロック番号「１－３」）が１番目のオーバーフロー・ブロック１３（ブロッグ番号「１－２」）に連結されていない状態にする（図３のＳ６）。

【００８５】

次に、（現用の）再編成ポインターＲＰＬＣが現用のロケーション・テーブルＬＣの３番目のエントリーを指すように更新し、（新規の）再編成ポインターＲＰＬＮが（新規の）ロケーション・テーブルＬＮの５番目のエントリーを指すように更新する。

次に、（現用の）再編成ポインターＲＰＬＣの２番目のエントリーと、そのエントリーで管理しているプライマリー・ブロック１２（図３ではブロック番号「

１」と表示されている）及びオーバーフロー・ブロック１３（図３ではブロック番号「１－２」と表示されている）、オーバーフロー・ブロック１４（図２ではブロック番号「１－３」と表示されている）の排他を解除する。

次に、３番目のエントリー（ブロック番号「２」）の処理を行う。現用のロケーション・テーブルＬＣの３番目のエントリーと、そのエントリーで管理しているプライマリー・ブロック１２（ブロック番号「２」）及びオーバーフロー・ブロック１３（ブロック番号「２－２」）を排他する。

【００８６】

新規のロケーション・テーブルＬＮの５番目のエントリーには、プライマリー・ブロック１２（ブロッグ番号「２」）のアドレスとするので、現用のロケーション・テーブルＬＮのアドレスと同じ値が入る（図３のＳ７）。次に、新規のロケーション・テーブルＬＮの６番目のエントリーには、１番目のオーバーフロー・ブロック１３のアドレス（ブロック番号「２－２」）の主キー値の最小値及び最大値を入れる（図３のＳ８）。

次に、プライマリー・ブロック１２（ブロック番号「２」）中のオーバーフロー・ブロック・アドレスをリセットして、オーバーフロー・ブロック１３（ブロック番号「２－２」）がプライマリー・ブロック１２（ブロック番号「２」）に連結されていない状態にする（図３のＳ９）。

次に、（現用の）再編成ポインターＲＰＬＣが現用のロケーション・テーブルＬＣの４番目のエントリーを指すように更新し、（新規の）再編成ポインターＲＰＬＮが（新規の）ロケーション・テーブルＬＮの６番目のエントリーを指すように更新する。

【００８７】

次に、（現用の）再編成ポインターＲＰＬＣの４番目のエントリーと、そのエントリーで管理しているプライマリー・ブロック１２（図３ではブロック番号「２」と表示されている）及びオーバーフロー・ブロック１３（図３ではブロック番号「２－２」と表示されている）の排他を解除する。

次に、再編成ポインターＲＰＬＣの４番目のエントリーと、そのエントリーで管理しているプライマリー・ブロック１２（のブロック番号「３」）及びオーバ

ーフロー・ブロックを排他する。この場合はオーバーフロー・ブロックが存在しないためプライマリー・ブロック１２（のブロック番号「３」）のみとなる。

【００８８】

次に、１番目のエントリー（ブロック番号「３」）を現用のロケーション・テーブルＬＣから新規のロケーション・テーブルＬＮに書き移す（図３のＳ１０）。その際に、１番目のエントリー（ブロック番号「３」）で管理しているブロック（１０のブロック番号「３」）に、オーバーフロー・ブロックが連結されているか否かを確認する。連結されていないので、（現用の）再編成ポインターＲＰＬＣ番目のエントリーの先頭を指し示すように変更する。また、（新規用の）再編成ポインターＲＰＬＮのアドレスを７番目のエントリーの先頭を指し示すように変更する。

【００８９】

（現用の）再編成ポインターＲＰＬＣの４番目のエントリーと、そのエントリーで管理しているプライマリー・ブロック（１０のブロック番号「３」）及びオーバーフロー・ブロックの排他を解除する。この場合はオーバーフロー・ブロックが存在しないためプライマリー・ブロックのみとなる。

このようにすることで、オーバーフロー・ブロックを、別の場所に書き換えることなく、再編成を行うことができる。

以下、同様に、５番目（ブロック番号５）のエントリーというように、順次再編成を行っていく。

図３は現用のロケーション・テーブルＬＣの４番目のエントリーまで、再編成が完了した状態を示している。

図３では（新規の）再編成ポインターＲＰＬＮの値は、ロケーション・テーブルＬＮの８番目（ブロック番号「７」）のエントリーの先頭を指す状態になっている。

【００９０】

[代替キー・テーブルのエントリーの形式による問題点とその解決法]

前述の再編成を行う場合に、代替キー・テーブルのエントリーを書き換える必要がある。これは、代替キー・テーブルのエントリーの形式によって異なる。

【００９１】

＜代替キー・テーブルのエントリー形式の違いによる方式との差＞

上記「データ格納検索方式」では、代替キー・テーブル（図１の符号１１Ａ，１１Ｂ，１１Ｃ参照）のエントリーは代替キー、そのキー値のレコードが格納されているブロックの物理アドレス、そのキー値のレコードの主キーからなる、としてあり、また、図では、代替キー、そのキー値のレコードが格納されているブロックのブロック番号、そのキー値のレコードの主キーからなる方式も示されている。

【００９２】

まず、代替キー・テーブルのエントリーが、代替キー、そのキー値のレコードが格納されているブロックのブロック番号、そのキー値のレコードの主キーからなる場合に関して述べる。

前述した操作の後、旧１番目と旧２番目のオーバーフロー・ブロックに格納されていたレコードの、代替キー・テーブルのエントリーの変更を行う。これは、代替キー・テーブルのエントリーに、レコードが格納されているブロックの番号を保持している場合には、再編成前は、プライマリー・ブロック、オーバーフロー・ブロックとも、ブロック番号が１であったが、再編成により、旧プライマリー・ブロックはブロック番号が１、旧１番目のオーバーフロー・ブロックはブロック番号が２、旧２番目のオーバーフロー・ブロックはブロック番号が３となる。この変更を、必要な代替キー・テーブルのエントリーに反映する。

この操作は、一時点での再編成対象レコードの代替キー・テーブルのエントリーを対象にすることになるため、オーバーフロー・ブロックの切り離しに比べて、時間がかかることに留意が必要である。

【００９３】

＜代替キー・テーブルのエントリーの新しい形式追加＞

形式を採用するのが、再編成には有利である。つまり、代替キー・テーブルの代替キー・テーブルのエントリー中に、ブロック番号を保持する。このため、代替キー・テーブルのエントリー中に、ブロック番号を保持エントリーを、代替キー値とそのキー値のレコードの主キー値というようにすることで、再編成に伴う

代替キー・テーブルのエントリーの書き換えが不要となり、再編成に要する負荷を削減することが可能である。しかしながら、この方法を採用した場合には、代替キーを用いた検索では、代替キー・テーブルを検索して主キー値を求めた後に、その主キー値でロケーション・テーブルの検索を行わないと、目的のブロックを検出できないため、検索の時間がかかることになる。

【００９４】

次に、代替キー・テーブルのエントリーが代替キー、そのキー値のレコードが格納されているブロックの物理アドレス、そのキー値のレコードの主キーからなる場合に関して説明を行う。

この形式の場合には、代替キーで検索を行った場合に、直接、ブロックを検索できる他、再編成の際にブロックのアドレスが変わらないために、再編成時の負荷が、ブロック番号を使用する場合に比較して低減するという利点がある。

このような利点がある一方で、次のような弱点もあるので、この方式の使用には十分な検討が必要である。まず、フラグメンテーションの解消の場合に、レコードが格納されるブロックが変わる場合には、そのレコードに関連する代替キー・エントリーのブロック・アドレスを書き換える必要があり、相応の負荷が発生することになる。レコードが格納されるブロックが変わるのは、レコードの挿入によってオーバーフロー・ブロックが追加され、レコードが新たにオーバーフロー・ブロックに移動される場合にも発生する。

【００９５】

次に、後述するように、排他の順序が他の場合と異なるため、デッドロックを引き起こす可能性が増大する。

また、後述するように、国内優先「データ・バックアップ・リカバ リー方式」を使用して、リカバリーを行う場合に制限が発生する。

ロケーション・テーブルの再編成の頻度や、リカバリーの容易さを考慮して、何れの形式とするかを選択することが好ましい。

【００９６】

[ロケーション・テーブル及びブロックの再編成：オーバーフローの解消の例外について]

オーバーフロー・ブロックの解消に関して述べたが、解消ができない場合に関して説明する。これは、スパンド・レコードが存在する場合である。スパンド・レコードとは、ブロックよりも大きなレコードがあった場合に、ブロックに格納できる大きさに分割して複数のブロックに格納する、というもので、旧来から利用されてきた形式である。

【００９７】

スパンド・レコードは、一つのレコードを分割したものであるので、同じ主キーを持つことになり、複数のプライマリー・ブロックに格納されることは無く、必ず一つのプライマリー・ブロックと１つ以上のオーバーフロー・ブロック、または、複数のオーバーフロー・ブロックに格納される。複数のオーバーフロー・ブロックに格納されるとは、プライマリー・ブロックとその直後のオーバーフロー・ブロックには既にレコードが格納されており、オーバーフロー・ブロックの途中からスパンド・レコードの格納が開始されるような場合である。

スパンド・レコードが格納されている場合は、ブロックにその情報を保持するのが分かりやすい。スパンド・レコードの先頭部分か、中間部分か、最後尾部分か、といった情報を保持するのである。それ以外は、通常のレコードを格納する場合と同様である。

この場合の、オーバーフロー・ブロックの解消は、通常のレコードの部分に関しては、解消を行えるが、スパンド・レコードが格納されるブロックに関しては、必ずオーバーフロー・ブロックを伴った形式となるため、オーバーフロー・ブロックの解消はできない。このような場合には、再編成後の情報として、スパンド・レコードに関するものを出力することが好ましい。

【００９８】

［ロケーション・テーブル及びブロックの再編成：フラグメンテーションの解消について］

上記では、オーバーフロー・ブロックを解消する方法に関して述べたが、オーバーフローと同様にフラグメンテーションも、効率上では大きな問題であった。これを解消するためには、オーバーフロー・ブロックを解消する再編成と似た動作で実現できることを、図４を参照しながら説明する。

ここに、図４は、本第１の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいて、フラグメンテーションを解消する再編成の動作を説明するための図である。

図４において、プライマリー・ブロック１２，１３，１４のブロック番号「０」、「１」、「１－２」、「１－３」、「２」、「２－２」には、適度にレコードが格納されており、オーバーフロー・ブロックの解消を前述の方法で完了したとする。以下の説明で使用するブロック番号は、現用のロケーション・テーブルＬＣに基づく番号を使用する。新規のロケーション・テーブルＬＮに基づく番号を使用する場合には、その旨を記述する。

【００９９】

プライマリー・ブロック１２のブロック番号「３」には、格納容量の３０％のレコードが格納されている。プライマリー・ブロック１２のブロック番号「４」には格納容量の４０％のレコードが格納されている。プライマリー・ブロック１２のブロック番号「５」は７０％、オーバーフロー・ブロック１３のブロック番号「５－２」には６０％、オーバーフロー・ブロック１３のブロック番号「６」には７０％のレコードが格納されている。また、再編成後の各ブロックの適正な初期格納率は９０％とする。これは、再編成後に挿入レコードが発生した場合に、その都度オーバーフロー・ブロックを発生させないために設けることができる。

【０１００】

再編成システムは、現用のロケーション・テーブルＬＣの再編成ポインターＲＰＬＣが指している４番目のエントリーが指しているブロック番号「３」の再編成を行おうとする。しかしながら、格納率（ブロックの容量に対する、そのブロックに格納されているレコードの容量の割合）が３０％であるため、適正な初期格納率を満たしていない。このため、次の現用のプライマリー・ブロック１２のブロック番号「４」を見る。このプライマリー・ブロック１２のブロック番号「４」も格納率が４０％であるため、２つのブロックを足しても適正な初期格納率（９０％）を下回ることになる。更に、プライマリー・ブロック１２のブロック番号「５」を見ると格納率は７０％であるため、適正な初期格納率９０％を上回

ることになる。

【０１０１】

このプライマリー・ブロック１２のブロック番号「３」に格納されているレコードはそのまま残し、プライマリー・ブロック１２のブロック番号「４」に入っていたレコードをプライマリー・ブロック１２のブロック番号「３」に移動する。これにより、プライマリー・ブロック１２のブロック番号「３」の格納率は７０％となる。更に、適正な初期格納率にするため、プライマリー・ブロック１２のブロック番号「５」に格納されているレコードの先頭の２０％分をプライマリー・ブロック１２のブロック番号「３」に移動すると同時に、プライマリー・ブロック１２のブロック番号「５」の残りのレコードの５０％分を前方に詰める（図４左側に移動させる）。この際に、移動されるレコードの代替キー・テーブルのエントリーを修正する方法に関しては、オーバーフロー・ブロックの解消で述べた方法と同様である。

【０１０２】

ブロックに格納されているレコードの主キー値の最小値と最大値に関する変更を行う。更に、現用のロケーション・テーブルＬＣ及び新規のロケーション・テーブルＬＮのエントリーに、ブロックの主キー値の最大値、最小値を持っている場合、新規のロケーション・テーブルＬＮのエントリーの主キー値の最小値と最大値の変更を行う。

プライマリー・ブロック１２のブロック番号「３」が完成したので、新規のロケーション・テーブルＬＮの７番目のアドレスを、ブロック番号「３」に書き換える（図４のＳ２０）。

再編成ポインターＲＰＬＮをロケーション・テーブルＬＮの８番目の先頭に移動する。この時点でプライマリー・ブロック１２のブロック番号「４」は使用されないブロックとなる（図４のＳ２１）。

【０１０３】

次に、プライマリー・ブロック１２のブロック番号「５」を操作するのであるが、再編成によりプライマリー・ブロック１２のブロック番号「５」は格納率が５０％となっている。オーバーフロー・ブロック１３のブロック番号「５－２」

は格納率が６０％であるので、オーバーフロー・ブロック１３のブロック番号「５－２」のレコードの内、先頭から３０％のレコードをプライマリー・ブロック１２のブロック番号「５」側に移動し、同時にオーバーフロー・ブロック１３のブロック番号「５－２」に残っているレコードを前方に詰める（レコードを図示左側に移動させる）。更に、プライマリー・ブロック１２のブロック番号「５」がオーバーフロー・ブロック１３のブロック番号「５－２」をオーバーフロー・ブロックとして連結しているのを切断する（図４のＳ２２）。これは、プライマリー・ブロック１２のブロック番号「５」のオーバーフロー・ブロック・アドレスを特定の値（例えばゼロ）にセットする。また、移動されるレコードの代替キー・テーブルのエントリーを修正する方法に関しては、オーバーフロー・ブロックの解消で述べた方法と同様である。

【０１０４】

プライマリー・ブロック１２のブロック番号「５」は完成したので、ロケーション・テーブルＬＮの８番目のアドレスをプライマリー・ブロック１２のブロック番号「５」に書き換える（図４のＳ２３）。

ついで、再編成ポインターＲＰＬＮをロケーション・テーブルＬＮの９番目の先頭に移動する。

次に、オーバーフロー・ブロック１３だったブロック番号「５－２」の格納率が３０％であるので、次のプライマリー・ブロック１２のブロック番号「６」を見ると格納率が６０％であるので、次のプライマリー・ブロック１２のブロック番号「６」のレコードのすべてをオーバーフロー・ブロック１３だったブロック番号「５－２」に移動する。オーバーフロー・ブロック１３だったブロック番号「５－２」の格納率が適正なものに完成したので、新規のロケーション・テーブルＬＮの８番目のアドレスをオーバーフロー・ブロック１３だったブロック番号「５－２」に書き換える（図４のＳ２４）。再編成ポインターＲＰＬＮをロケーション・テーブルＬＮの９番目の先頭に移動する。この時点でプライマリー・ブロック１２のブロック番号「６」は使用されないブロックとなる（図４のＳ２５）。

【０１０５】

[適正な初期格納率の確保]

また、フラグメンテーションとは逆に、適正な初期格納率を確保するために、ブロックの追加が必要な場合も発生する。これは、例えば、すべてのブロックの格納率が１００％であった場合に、適正な初期格納率を９０％とすると、一時点で９個のブロックを再編成の対象にし、１つのブロックを追加して１０個のブロックに対して、レコードを９０％ずつ格納する、というような 形態である。

適正な初期格納率の確保を厳密に行おうとする、ブロックとレコードの大きさの関係で不可能な場合や、一時点でかなり多数のブロックを再編成の対象にしなければならなくなる可能性があるため、排他範囲が広がり、システムの稼動に悪影響を与える可能性がある。

【０１０６】

＜複数の初期格納率＞

このような状態が発生することを防止する上で、適正な初期格納率を、例えば、８５％から９０％など、複数の値を用いることが、運用上は好ましい。この場合、適正な初期格納率が、８５％から９０％の範囲に入っていればよい、とするものである。

また、適正な初期格納率に関しては、ブロック毎のオーバーフロー・ブロックの発生状況によって、ブロック毎に定めることも可能である。この場合の適正な初期格納率は、ロケーション・テーブルのエントリーに項目を追加して、そこに記載しておくか、ブロック内に項目を設けて記載しておくことで実現できる。

この適正な初期格納率の確保を行う場合にも、レコードが元のブロックから別のブロックに書き換えられることが発生する。このように、書き換え対象となるレコードに関係する代替キー・テーブルのエントリーを書き換える必要性がある。

【０１０７】

[実際の再編成とデッドロックの防止]

実際に再編成を行う場合には、オーバーフロー・ブロックの解消と、フラグメンテーションの解消、及び、適正な初期格納率の確保を同時に実行することになるので、上記の３つの方式を組み合わせて使用する。

原理的には以上のような方法で実現が可能であるが、ブロックを順次読んでいき、適正な初期格納率のブロックを作成していくと、当初から排他の範囲が確定せず、順次に排他していくために、デッドロックが発生する可能性が増加する。デッドロックの発生を防ぐためには、次のような方法が有効である。

「デッドロック発生を防止する第一の方法」は、まず、排他を掛けずに次々とブロックを読みだし、レコードの格納率を調べて、複数のブロックを組み合わせて、適正な大きさとなるように計算をした上で、排他範囲を決定する方法である。

「デッドロック発生を防止する第二の方法」は、次の様なものである。上述した「データ格納検索方式」においては、ロケーション・テーブルのエントリーは、ブロック番号とブロックのアドレスの他、ブロックに格納されているレコードの主キー値の最小と最大の両方、または何れか一方を保持できるとしてある。

【０１０８】

＜ロケーション・テーブルのエントリーへの項目追加＞

これらの情報に加えて、ブロック中のレコードの格納率若しくは、レコードが占めている領域のバイト数を、ブロックに連結されているオーバーフロー・ブロックの数を、ロケーション・テーブルのエントリーに追加する。このようにすることで、第一の方法で述べた、ブロックを読むという作業を必要とせず、ロケーション・テーブルを読むだけで作業が行えることになる。しかしながら、この場合にはレコードの挿入や削除に伴って、ブロック中のレコードの格納率若しくは、レコードが占めている領域のバイト数が増減し、ロケーション・テーブルの書き換えが発生することになるため、レコードの発生状況に応じて、前述したデッドロック発生を防止する第一の方法と第二の方法のいずれかを選択することが好ましい。

【０１０９】

また、上記の情報はブロックの中に保持することも可能である。

このようにして、ブロック内の格納率を調べた後で、その時点での再編成範囲を確定し、その範囲のロケーション・テーブルのエントリーと、そのエントリーが指しているブロック及び、ブロックに連結されているオーバーフロー・ブロッ

クの排他を行う。

その後、再編成対象となるブロックとオーバーフロー・ブロックの数と格納可能容量を調べ、格納対象レコードの実際の容量を調べた上で、必要になるブロックが現用のブロックとオーバーフロー・ブロックの和と同数か、増加するか、減少するか、を調べる。この後、オーバーフロー・ブロックの解消や、フラグメンテーションの解消、適正な初期格納率の確保で説明したロジックを使用して、ブロックの再編成と、新規ロケーション・テーブルのエントリーの作成を行う。この場合には、再編成ポインターは、再編成の対象となったブロックの分だけ、一度に移動することになる。

【０１１０】

このように、一時点で、１個から数十個のブロックを対象として再編成を実施するのであるが、この再編成は、見かけ上、通常のデータ処理のトランザクションとして扱うことにより、通常のデータ処理との矛盾を発生させない。

また、この再編成を次々と、ロケーション・テーブルとブロックに対して実施することにより、それら全体の再編成を完了する。

【０１１１】

[再編成の完了]

ロケーション・テーブルの再編成が完了したことを認識する方法に関して説明する。現用のロケーション・テーブルＬＣには、そのロケーション・テーブルを使用している最終位置を示すための、ラスト・ポインターが設けてある。

ラスト・ポインターを設ける目的は、次のようなものである。ロケーション・テーブルは連続領域に確保することになっている。ロケーション・テーブルのエントリーが不足した場合には、最初のロケーション・テーブルとは連続しない領域に追加のロケーション・テーブルを設けて、それらを連続した領域であるかのようにアドレス変換してバイナリー・サーチする方法が「データ格納検索方式」では示されているが、不連続領域が多くなると、アドレス変換の負荷が増大するため、好ましくない。このため、ロケーション・テーブル用の領域は、予め十分に大きな領域を確保しておき、使用しているエントリーと、未使用のエントリーを区分するために、使用しているエントリーの次のエントリーのアドレスを指す

、ということになっている。

このように、ラスト・ポインターを設けることにより、現用のロケーション・テーブルＬＣの先頭アドレスとラスト・ポインターの間でバイナリー・サーチを実行することにより、現用のロケーション・テーブルＬＣに未使用エントリーが存在していても、主キーによるレコード検索が可能となる。

【０１１２】

＜再編成の完了の検出方法＞

このラスト・ポインターを指標として用いる。再編成を実行して行き、現用の再編成ポインターＲＰＬＣが指すアドレスが、ラスト・ポインターが指すアドレスと一致した場合に、ロケーション・テーブルとブロックの再編成が完了したことになる。尚、最終エントリーが指しているブロックに、オーバーフロー・ブロックが連結されている場合でも、それらの、オーバーフロー・ブロックの再編成が完了しない限り、現用の再編成ポインターＲＰＬＣの移動は行われないので、問題ない。

再編成が完了した場合には、現用の再編成ポインターＲＰＬＣは不要となるので、新規のロケーション・テーブルＬＮを現用のロケーション・テーブルとし、ロケーション・テーブルＬＣは消去してかまわない。

【０１１３】

［未使用ブロックの再利用］

このようにして、フラグメンテーションに対する再編成の方法に関して述べたが、図４の場合には、現用のロケーション・テーブルＬＣのプライマリー・ブロック１３のブロック番号「４」と、プライマリー・ブロック１３のブロック番号「６」が使用されなくなってしまった。このままでは、ブロック内のフラグメンテーションは解消されるが、全体的なフラグメンテーションは解消されない、ということになりかねない。このようなことを防止するために、つぎのような工夫を採用する。

【０１１４】

＜未使用ブロック・アロケーション・テーブル。開始位置ポインター、終了位置ポインター＞

図５及び図６は、本第１の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいて、全体的なフラグメンテーションを解消させる方法について説明するための図である。

データ及びデータベースの無停止自動再編成システムにおいて、全体的なフラグメンテーションを解消させる方法では、図５に示すように、未使用ブロック・アロケーション・テーブルＵＢＡＴというものを使用する。この未使用ブロック・アロケーション・テーブルＵＢＡＴは、図５に示すような形式のテーブルであって、ブロック１０において未使用ブロックのアドレスを格納しておくためのものである。また、データ及びデータベースの無停止自動再編成システムにおいて、全体的なフラグメンテーションを解消させる方法では、未使用ブロック・アロケーション・テーブルＵＢＡＴの開始位置を示すポインター（開始位置ポインターＮＡＢＰＳ）と終了位置を示すポインター（終了位置ポインターＮＡＢＰＥ）の２つを使用する。図５では、未使用ブロックが全く無い状態から、７つの未使用ブロックが発生した後の状態を示している。

【０１１５】

初期状態では、開始位置ポインターＮＡＢＰＳと終了位置ポインターＮＡＢＰＥとは、両方とも未使用ブロック・アロケーション・テーブルＵＢＡＴの先頭を指している。

ここで、未使用ブロックが発生すると、未使用ブロック・アロケーション・テーブルＵＢＡＴの終了位置ポインターＮＡＢＰＥの指しているエントリーに、ブロック１０の未使用ブロックのアドレスを登録し、終了位置ポインターＮＡＢＰＥを未使用ブロック・アロケーション・テーブルの次のエントリーを指すように書き換える。このような操作を順次おこなった結果、７つの未使用ブロックが発生した後の状態をしめすものが図５である。ここでは、終了位置ポインターＮＡＢＰＥは、図５に示すように、未使用ブロック・アロケーション・テーブルＵＢＡＴの８番目のエントリーを指している。

【０１１６】

次に未使用ブロックの再利用の方法について説明する。本第１の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいて、新た

にブロックを取得する必要が発生した場合（例えば、レコード追加で、ロケーション・テーブルのラスト・ポインターの次にプライマリー・ブロックを追加取得する場合と、オーバーフロー・ブロックを追加する場合）、新たな領域からブロックを取得するのではなく、未使用ブロック・アロケーション・テーブルＵＢＡＴを参照し、未使用ブロックが存在する場合には、未使用ブロック・アロケーション・テーブルＵＢＡＴに登録してあるブロックを優先して使用する。未使用ブロックを使用する方法は、開始位置ポインターＮＡＢＰＳが指しているエントリーのブロックを使用する。未使用ブロック・アロケーション・テーブルＵＢＡＴのエントリーには未使用ブロックのアドレスが入っているので、このアドレスを、ブロック追加の場合はロケーション・テーブル（図２ないし図４では、ロケーション・テーブルＬＣ及びＬＮ）に書込み、オーバーフロー・ブロックの場合は、そのブロックを管理するプライマリー・ブロックまたはオーバーフロー・ブロックのポインターに、そのアドレスを書き込む。その後、開始位置ポインターＮＡＢＰＳの内容を未使用ブロック・アロケーション・テーブルの次のエントリーを指すように書き換える。このような書き換え処理が２回実行され、２つの未使用ブロックが使用された直後の状態を示したものが図５である。

【０１１７】

この未使用ブロック・アロケーション・テーブルＵＢＡＴは、循環使用が可能である。未使用ブロックが発生すると、終了位置ポインターＮＡＢＰＥの位置は、未使用ブロック・アロケーション・テーブルＵＢＡＴの後方（図示下側方向）に動いていくが、一方で、未使用ブロックが使用されると、開始位置ポインターＮＡＢＰＳの位置も未使用ブロック・アロケーション・テーブルＵＢＡＴの後方（図示下側方向）にずれていくので、終了位置ポインターＮＡＢＰＥが開始位置ポインターＮＡＢＰＳを追い越さない限り、一つのテーブルを循環して使用する。つまり、本第１の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいて、終了位置ポインターＮＡＢＰＥが未使用ブロック・アロケーション・テーブルＵＢＡＴの最後（図示最下部）の位置に達したときにはまた、未使用ブロック・アロケーション・テーブルＵＢＡＴの先頭（図示最上部）に終了位置ポインターＮＡＢＰＥを戻し、未使用ブロック・アロケーション

・テーブルＵＢＡＴを再使用することが可能である。

【０１１８】

[再編成中のデータベースのアクセス]

次に、本第１の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいて、再編成中の、データの検索・読み書きに関して、再編成を行っているときにも、データの検索・読み書きが可能であることを、図７を参照しながら述べる。

付言すれば、再編成中にデータの検索・読み書きが可能であるということが、「データ格納・検索システム」の稼動を行ったまま、つまり、システムを停止することなく、再編成を実施できるということになる。

ここに、図７は、本第１の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいて、再編成中のデータの検索・読み書き動作について説明するための図である。

このデータ及びデータベースの無停止自動再編成システムにおいて、まず、再編成を行っていない状態での主キー値による検索では、現用のロケーション・テーブルＬＣを用いてバイナリー・サーチを行っている。

このデータ及びデータベースの無停止自動再編成システムにおいては、再編成を行っている場合には、ターゲット・キー値（検索の対象となるキー値）が再編成ポインターＲＰＬＣが指している主キー値より小さい（図示上方にある）か、大きいか（図示下方にある）の判定を行う。これは、現用のロケーション・テーブルＬＣが主キー値の順に並んでいることから、再編成ポインターＲＰＬＣの示しているエントリーのキー値とターゲット・キー値の比較を行えばよい。

ターゲット・キー値が、再編成ポインターＲＰＬＣが指しているエントリーの主キー値の最小値より小さい場合は、再編成ポインターＲＰＬＣより上方（アドレスが小さい）に、目的とするエントリーが存在することになる。この場合は、新規のロケーション・テーブルＬＮを使用して、新規のロケーション・テーブルＬＮの先頭アドレスと再編成ポインターＲＰＬＮ間で（検索対象領域１０１内を）バイナリー・サーチを行う。バイナリー・サーチの結果、目的のエントリーが指しているブロックの中のレコードを調べて、目的のレコードが存在するか否かを

検出する。

【０１１９】

ターゲット・キー値が現用の再編成ポインターＲＰＬＣ０の指しているエントリーの主キー値の最小値と等しいか、または、大きい場合には、目的とするエントリーは、現用の再編成ポインターＲＰＬＣより下方（ＲＰＬＣ０が指しているエントリー、または、それよりアドレスが大きい）に存在することになる。この場合は、現用のロケーション・テーブルＬＣを使用して、現用の再編成ポインターＲＰＬＣと現用のロケーション・テーブルＬＣのラスト・ポインター間（検索対象領域１０２内）のエントリーに対してバイナリーサーチを行う。

【０１２０】

このように、本第１の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムでは、現用の再編成ポインターＲＰＬＣを用いて、ターゲット・キー値と比較し、現用の再編成ポインターＲＰＬＣの指している主キー値より小さい（図示上方）か、大きいか（図示下方）かを判別し、小さい（図示上方）ときにはロケーション・テーブルＬＮを、大きいか等しい（図示下方）ときにはロケーション・テーブルＬＣを、それぞれバイナリー・サーチすることにより、目的のエントリーの検索が確実に実行できるので、目的とする主キー値を持つレコードの検索が可能となる。

【０１２１】

また、本第１の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムでは、目的とする主キー値を持つレコードが含まれるブロックが、正に再編成中で、排他されている場合には排他待ちとなり、その排他が解除されるまでは、当該ブロックのアクセスはできないことになるが、このような状態は、通常のアクセスでレコードの更新や挿入、削除が行われる場合と同様であるので、まったく問題はない。言い換えれば、排他中のブロックに対しての要求は、排他待ちになるが、そのブロックに対する再編成が完了して、排他が解除されれば、処理が可能となる。

【０１２２】

[代替キーによる検索について]

次に、本第１の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいて、代替キーによる検索について図１ないし図４を参照して説明する。

上記の説明では、ロケーション・テーブルとブロックの再編成を行っている場合に、主キーによる検索の事例に関して説明したが、代替キーによる検索では、以下のようになる。代替キーによる検索では、代替キー・テーブル（図１を参照すれば、符号１１Ａ，１１Ｂ，１１Ｃ）、または、後述するように代替キー・ロケーション・テーブルを使用して対象代替キー・エントリーを検索する。

当該代替キー・エントリーが見つかったら、その内容に基づいて、レコードの検索を行うことになる。代替キー・エントリーの形式は、前述したように３種類ある。（ｉ）ブロック番号を保持する場合、(ii)ブロック番号を保持しない場合、(iii)ブロック・アドレスを保持する場合である。

【０１２３】

第(ii)項のブロック番号を保持しない場合は、主キーによって、ロケーション・テーブルのバイナリーサーチを行うので、上記、再編成中のデータベースのアクセスの説明と全く同様である。ブロック・アドレスを保持している場合には、再編成によってブロック・アドレスが変更されないので、そのアドレスのブロックが該当ブロックであることが分かる。ところが、ブロック番号の場合には、再編成によって更新されるので、現用のロケーション・テーブルの番号であるか、新規のロケーション・テーブルの番号であるかの識別が必要となる。

【０１２４】

この識別は以下のように行う。

現用の再編成ポインターＲＰＬＣ（図３、図４参照)が指しているブロック番号が新規の再編成ポインターＲＰＬＮが指しているブロック番号と同じである場合は次のようになる。検索対象となるブロック番号が、現用の再編成ポインターＲＰＬＣが指しているブロック番号以上の場合は現用のロケーション・テーブルＬＣを使用し、未満の場合は新規のロケーション・テーブルＬＮを使用する。

【０１２５】

現用の再編成ポインターＲＰＬＣが指しているブロック番号が新規の再編成ポ

インターＲＰＬＮが指しているブロック番号より大きい場合は次の様になる。現用の再編成ポインターＲＰＬＣが指しているブロック番号と新規の再編成ポインターＲＰＬＮが指しているブロック番号の間のブロックが、検索対象となることは有り得ない。何故ならば、現用のロケーション・テーブルＬＣでは編成ポインターＲＰＬＣの位置まで再編成が済んでおり、新規のロケーション・テーブルＬＮでは、新規の再編成ポインターＲＰＬＮより上位の部分は再編成を行っていないため、新規の再編成ポインターＲＰＬＮが指しているブロックよりブロック番号が大きいブロックが対象とはならないためである。

【０１２６】

つまり、新規の再編成ポインターＲＰＬＮが指しているブロック番号より小さいブロック番号が検索対象となった場合には、新規のロケーション・テーブルＬＮを使用する。再編成ポインターＲＰＬＣ以上のブロック番号が検索対象となった場合には、現用のロケーション・テーブルＬＣを使用する。

現用の再編成ポインターＲＰＬＣが指しているブロック番号が新規の再編成ポインターＲＰＬＮが指しているブロック番号より小さい場合は次の様になる。

検索対象となるブロック番号が、再編成ポインターＲＰＬＣが指しているブロック番号より小さい場合は、新規のロケーション・テーブルＬＮを使用する。検索対象となるブロック番号が、再編成ポインターＲＰＬＮが指しているブロック番号より大きい場合は、現用のロケーション・テーブルＬＣを使用する。

【０１２７】

検索対象となるブロック番号が、現用の再編成ポインターＲＰＬＣが指しているブロック番号以上で新規の再編成ポインターＲＰＬＮが指しているブロック番号未満の場合には、ブロック番号だけでは何れのロケーション・テーブルが対象であるか、一意に特定できないため、次の様に行う。

【０１２８】

代替キー・テーブルの検索の結果、求められた代替キー・エントリーには、主キー値が含まれている。この主キー値と再編成ポインターＲＰＬＮが指しているロケーション・テーブルのエントリーの主キー値を比較し、代替キー・エントリーの主キー値が再編成ポインターＲＰＬＮの主キー値未満である場合には新規の

ロケーション・テーブルＬＮを使用する。代替キー・エントリーの主キー値が新規の再編成ポインターＲＰＬＮの主キー値以上である場合には現用のロケーション・テーブルＬＣを使用する。

【０１２９】

上記では、レコード検索の場合に関して述べたが、これを応用することで、レコードの挿入、更新、削除を行うことも可能である。

まず、挿入の場合に関して説明する。レコードを挿入する場合は、挿入対象レコードの主キー値によって、どのブロックのどの位置に挿入するかを決定する必要がある。これは、上述したデータ格納・検索方式では、レコードは主キー値の順にブロックに格納し、更にブロック間では、あるブロックより前にあるブロックに格納されているレコードの主キー値は、あるブロックに格納されているレコードの主キー値より小さい、という格納方式が定めてあるからである。

【０１３０】

検索と全く同じ方法によって、挿入対象レコードの挿入すべきブロックを求め、そのブロック内を検索することにより、レコードの挿入位置を求める。ここで、ロケーション・テーブルの該当エントリーと、挿入対象ブロックの排他を行う。検索の場合の説明で記述したように、その対象ブロックが正に再編成中である場合には、排他されているので、排他待ちになるが、再編成の排他が解除されれば、排他を行って以下の動作に入れる。

【０１３１】

そのブロックにオーバーフロー・ブロックが連結されていない場合を、まず、説明する。

そのプライマリー・ブロックに挿入対象レコードを格納するだけの空きスペースが存在する場合は、挿入する位置以降のレコードを、挿入対象レコードの長さ分だけ後方に移動し、空いた領域に挿入対象レコードを書き込む。

そのプライマリー・ブロックに挿入対象レコードを格納するだけの空きスペースが無い場合には、新たにオーバーフロー・ブロックをそのプライマリー・ブロックに連結し、挿入対象レコードを格納した後の格納容量を計算し、プライマリー・ブロックに適正な初期格納率が確保されるように、必要なレコードをオーバ

ーフロー・ブロックに移動し、挿入対象レコードを、プライマリー・ブロックまたはオーバーフロー・ブロックの適切な位置に格納する。

そのブロックにオーバーフロー・ブロックが連結されている場合は、プライマリー・ブロックとオーバーフロー・ブロックを一体としてみなして、上述の操作を行う。

【０１３２】

次に、代替キーの追加を行う。挿入対象レコードに３種類（Ａ，Ｂ，Ｃ）の代替キーが存在する場合、その種類毎に、代替キー・テーブルの検索を行う。代替キー・テーブルＡの場合に関して説明する。再編成中の代替キーによる検索の方法に関しては、代替キー・テーブルの再編成で詳細に説明するので、ここでは省略する。代替キー・エントリーを挿入する代替キー・ブロックを検索したら、代替キー・ブロック内で代替キー・エントリーを挿入する位置を決定する。ここで、代替キー・ブロックとその代替キー・ブロックに代替キー・オーバーフロー・ブロックが連結されている場合には、その代替キー・オーバーフロー・ブロックの排他を行う。更に、代替キー・ロケーション・テーブルを使用している場合には、代替キー・ロケーション・テーブルの当該エントリーも排他を行う。その対象代替キー・ブロックが正に再編成中である場合（代替キー・テーブルの再編成に関しては後述する）には、排他されているので、排他待ちになるが、再編成の排他が解除されれば、排他を行って以下の動作に入れる。

【０１３３】

その代替キー・ブロックに代替キー・オーバーフロー・ブロックが連結されていない場合を、まず、説明する。

その代替キー・ブロックに挿入対象代替キー・エントリーを格納するだけの空きスペースが存在する場合は、挿入する位置以降の代替キー・エントリーを、挿入対象代替キー・エントリーの長さ分だけ後方に移動し、空いた領域に挿入対象代替キー・エントリーを書き込む。

【０１３４】

その代替キー・ブロックに挿入対象代替キー・エントリーを格納するだけの空きスペースが無い場合には、新たに代替キー・オーバーフロー・ブロックをその

代替キー・ブロックに連結し、挿入対象代替キー・エントリーを格納した後の格納容量を計算し、代替キー・ブロックに適正な初期格納率が確保されるように、必要な代替キー・エントリーを代替キー・オーバーフロー・ブロックに移動し、挿入対象代替キー・エントリーを、代替キー・ブロックまたは代替キー・オーバーフロー・ブロックの適切な位置に格納する。

そのブロックに代替キー・オーバーフロー・ブロックが連結されている場合は、代替キー・ブロックと第値キー・オーバーフロー・ブロックを一体としてみなして、上述の操作を行う。

代替キーＢ，Ｃに関しても全く同様な操作を行う。

以上の一連の操作を行うことで、レコードの挿入が完了することになる。ここで、ロケーション・テーブルの当該エントリー、当該ブロックと当該オーバーフロー・ブロック、当該代替キー・ブロック、当該代替キー・オーバーフロー・ブロックの排他を解除する。

【０１３５】

次に、レコードの更新の場合を説明する。

レコードの更新でも、同様に、まず、レコードを検索する必要がある。前記検索の場合の方式を用いて検索を行う。検索によって見つかった後、ロケーション・テーブルの当該エントリーと当該ブロックの排他を行う。その後、対象レコードの更新を行えばよい。レコード長が変化せず、代替キーが変更されない場合には、それで更新は完了するので、ロケーション・テーブルとブロックの排他を解除する。

更新によって、レコード長が変動する場合は、挿入の場合と似た操作になる。レコード長が長くなる場合には、長くなる分の格納スペースがあるか否かを判別する。格納スペースがある場合には、格納位置以降のレコードを必要なバイト数だけ後方に移動し、以前に対象レコードが占めていた格納場所と、新たに確保した格納スペースを合わせて、更新されたレコードを格納する。

【０１３６】

空きスペースが無い場合には、オーバーフロー・ブロックを連結し、レコードをオーバーフロー・ブロックに移動するが、それは、挿入の場合と同様である。

代替キー値が変更された場合には、代替キー・テーブルの変更が必要になる。代替キー値の変更というのは、旧代替キー値のエントリーの削除と新代替キー値のエントリーの追加を行うことになる。

代替キー・エントリーの追加は、レコード挿入の場合と同様である。代替キー・エントリーの削除は、該当する代替キー・ブロックを検索して、削除対象代替キー・エントリーを見つける。代替キー・ブロックが見つかった時点で、代替キー・ブロックとその代替キー・ブロックに代替キー・オーバーフロー・ブロックが連結されている場合には、その代替キー・オーバーフロー・ブロックの排他を行う。更に、代替キー・ロケーション・テーブルを使用している場合には、代替キー・ロケーション・テーブルの当該エントリーも排他を行う。

【０１３７】

削除対象エントリーを削除し、それ以降の代替キー・エントリーを代替キー・ブロック内で前方に移動する。削除対象代替キー・エントリーが代替キー・オーバーフロー・ブロックに格納されている場合には、代替キー・オーバーフロー・ブロック内で代替キー・エントリーの移動を行う。代替キー・ブロックに代替キー・オーバーフロー・ブロックが連結されている場合に、代替キー・ブロック内の代替キー・エントリーを削除する場合、連結されている代替キー・オーバーフロー・ブロック内の一部の代替キー・エントリーを代替キー・ブロックに移動してもよいが、移動しなくとも動作に問題は生じないので、移動しなくて良い。これは、代替キー・オーバーフロー・ブロック内に対象代替キー・エントリーが存在する場合も同様である。

変更された代替キーが複数ある場合には、上記の動作を必要な代替キー・テーブルに関して実行する。

以上の一連の操作を行うことで、レコード更新に関する操作が完了するので、掛けた排他を解除する。

【０１３８】

次に、レコードの削除の場合を説明する。削除の場合も、まず、削除対象レコードを検索する必要がある。レコードが見つかったら、ロケーション・テーブルの当該エントリーと当該ブロックに対して排他を掛ける。

次に、削除対象レコードの削除を行い、そのレコード以降のレコードを削除対象レコードが占めていた領域の分だけ、前方に移動する。プライマリー・ブロックにオーバーフロー・ブロックが連結されている場合に、プライマリー・ブロック内のレコードを削除する場合、連結されているオーバーフロー・ブロック内のレコードの一部をプライマリー・ブロックに移動してもよいが、移動しなくとも動作に問題は生じないので、移動しなくて良い。これは、オーバーフロー・ブロック内に対象レコードが存在する場合も同様である。

【０１３９】

以上のように、再編成中であっても、レコードの検索、挿入、更新、削除が行える。レコードの追加は、挿入の一変形例であるので、挿入と同様に行える。

【０１４０】

[検索中に再編成が進行した場合への対処]

図８は、本第１の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいて、検索中に再編成が進行した場合への対処する動作について説明するための図である。

本第１の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムでは、上述したように再編成ポインターを使用し、ターゲット・キー値と比較することで、現用のロケーション・テーブルＬＣ、新規のロケーション・テーブルＬＮの何れかを使用するかを決定することにより、再編成中でもレコードの呼び出しが可能であることを説明した。

【０１４１】

しかしながら、図８に示すように、現用のロケーション・テーブルＬＣを用いて、主キーの検索を行っている最中に再編成が進んでしまい、検索開始時の再編成ポインターＲＰＬＣの位置（図８のＳ３１）と、検索終了時の再編成ポインターＲＰＬＣの位置（図８のＳ３２）とが異なっていて、その範囲のブロックが検索対象となった場合には、オーバーフロー・ブロックの連結が、既に切れているために、実際には存在するレコードが、存在しないことになってしまう可能性がある。図８では、プライマリー・ブロック１２のブロック番号「５」に連結されていたオーバーフロー・ブロック１３（ブロック番号「５－２」）が切り離され

てしまっている。このままでは、不安定な動作を引き起こすことになり、使用できないという不都合が発生する。

【０１４２】

このような不都合は、次のような工夫を行うことにより、問題なく検索することが可能である。ターゲット・キー値と再編成ポインターによって、検索対象となるロケーション・テーブルが現用のロケーション・テーブルＬＣであるか、新規のロケーション・テーブルＬＮであるかの判定を行う。使用するロケーション・テーブルが新規のロケーション・テーブルＬＮである場合には、再編成ポインターＲＰＬＮが検索開始時よりも進んでも、問題は発生しない。問題は現用のロケーション・テーブルＬＣを対象とした場合である。現用のロケーション・テーブルＬＣをサーチ対象とした場合には、再編成ポインターＲＰＬＮが検索開始時よりも進んでいると、工夫をしないと、問題が発生する。

【０１４３】

＜再編成が進行した場合のアクセス方法＞

検索を開始する前に、再編成中であれば、現用の再編成ポインターＲＰＬＣの値と新規の再編成ポインターＲＰＬＮの値をメモリーの所定のエリアに保存しておく。これを、Ｓ－ＲＰＬＣと、Ｓ－ＲＰＬＮとする。また、新規のロケーション・テーブルＬＮの検索を完了した時点の再編成ポインターＲＰＬＮの値を、Ｅ－ＲＰＬＣに保存する。

【０１４４】

また、現用のロケーション・テーブルＬＣの検索を完了した時点で、その時点の再編成ポインターＲＰＬＣの値（これを、Ｅ－ＲＰＬＣとする）とＳ－ＲＰＬＣの値を比較する。この値が異なっていれば、検索中に再編成が進行したことを意味している。この場合には、検索対象ブロックが何処であるかを判定する。判定した結果が、Ｓ－ＲＰＬＣとＥ－ＲＰＬＣの間に対象ブロックがあるばあいには、前述したように、レコードが検索できない可能性がある。この場合には、新規ロケーション・テーブルＬＮを使用して、Ｓ－ＲＰＬＮとＥ－ＲＰＬＮの間に対してバイナリー・サーチを行う。ここで、レコードが発見できれば、レコード有り、発見できなければレコード無しとなる。このようにすることで、確実にレ

コードの検索が行えるので、存在するレコードが存在しないとなってしまう事象

以上で、説明したように、再編中であってもレコードの読み出しが可能である。

【０１４５】

[レコードの挿入・更新の場合]

呼び出しに関して述べたが、レコード挿入の場合も、レコードを挿入するブロックを決定するためには、バイナリー・サーチでブロックを見付けて、その後、当該ブロックに対してレコードを挿入する必要があり、呼び出しと同様の動作となる。

また、データの更新は、一旦、レコードを読み出した後に、更新を行い、格納する、という順序に従って行うことになるため、レコードの呼び出しで示した方式を使用するが、主キー値の変更の場合には、次のようになる。

レコードの主キー値が変更される場合には、レコードの格納場所を変更する必要がある。何故ならば、ブロックの中には、主キー値の順序でレコードが並ぶことになっており、その範囲を外れるレコードが存在すると、ロケーション・テーブルで検索が不能になってしまうからである。このため、主キー値を変更する場合には、現レコードを一旦削除し、その後、新しいレコードを、その主キー値に基づいて格納されるべきブロックを探して、そこに挿入する。この方法は、旧来のデータベースで一般的に使用されている方法である。

これ以外は、レコード呼び出しと同様の方法で、ブロックを探して書き込むことが可能である。

【０１４６】

[再編成の中断と再開]

以上で説明したように、ロケーション・テーブルの再編成中であっても、主キー値による、レコードの検索（読み込み）や、更新、レコードの挿入、追加、削除が可能であることを説明した。つまり、再編成がどの時点、言い換えれば、ロケーション・テーブルのどのエントリー、まで進行していても、レコードのアクセスが可能であることはいうまでもない。

【０１４７】

＜再編成の中断と再開＞

これに基づけば、一時的に再編成を中断しても、レコードのアクセスは問題なく実行できることが分かる。勿論、中断は、あるまとまりのブロックに対する再編成が完了した後に行われることは当然である。

本第１の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいて、再開は、中断した時点で、現用の再編成ポインターＲＰＬＣと新規の再編成ポインターＲＰＬＮが示している、現用のロケーション・テーブルＬＣと新規のロケーション・テーブルＬＮのエントリーから再編成を実施すればよい。

この機能により、プライマリー・システムの負荷が増加した場合には、再編成を中断して、データ処理に能力を振り分け、データ処理の負荷が減少してきたら、再編成を再開することが可能となるので、再編成の実施に当たって、プライマリー・システムの負荷や運転状況に関して事前予測を行い、一定の時間を予め確保する必要がない。

【０１４８】

［オーバーフロー・ブロックの形式］

上述した「データ格納検索方式」では、オーバーフロー・ブロックの形式に関しては、図に例示してある。これによれば、オーバーフロー・ブロックには、当該ブロック中のレコードの主キー値の最小値と最大値を保持しない形式となっている。また、プライマリー・ブロック中に、プライマリー・ブロック中のレコードの主キー値の最小値と最大値との他に、オーバーフロー・ブロック中のレコードの主キー値の最小値と最大値を保持するような図が例示されている。

【０１４９】

＜オーバーフロー・ブロックの形式＞

これらに対し、次のような形式を考案した。プライマリー・ブロックでは、当該プライマリー・ブロック中のレコードの主キー値の最小値と最大値を保持する。オーバーフロー・ブロックでは、同様に、当該オーバーフロー・ブロック中のレコードの主キー値の最小値と最大値を保持する。また、ロケーション・テーブルのエントリーでは、当該エントリーが管理するプライマリー・ブロックと、そ

のプライマリー・ブロックが管理するオーバーフロー・ブロックすべてに格納されているレコードの、主キー値の最小値と最大値の両方、若しくは一方を保持する。

【０１５０】

オーバーフロー・ブロックに主キーの最小値と最大値を保持しない場合には、その分だけ、領域をレコード格納に充当することが可能である。しかしながら、オーバーフロー・ブロックを再編成により解消して、プライマリー・ブロックにする場合に、主キーの最小値と最大値を保持することになるので、レコードが満杯に格納されていた場合には、主キーの最小値と最大値を保持するための領域が足りずに、オーバーフロー・ブロックが発生してしまう。また、一つのプライマリー・ブロックに対して、オーバーフロー・ブロックが多数、連結されている場合には、目的のレコードを探すために、オーバーフロー・ブロックの中のレコードを順次読んでいく必要があり、検索時間が余分に掛かる、という問題も発生する。これを、プライマリー・ブロックでは、当該プライマリー・ブロック中のレコードの主キー値の最小値と最大値を保持し、オーバーフロー・ブロックでは、同様に、当該オーバーフロー・ブロック中のレコードの主キー値の最小値と最大値を保持する、という形式とすることにより、次の効果がある。

【０１５１】

＜新しいオーバーフロー・ブロックの形式＞

以上説明したように、本第１の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムによれば、再編成により、オーバーフロー・ブロックをプライマリー・ブロックに変更した場合に、ブロックの形式変更がないので、主キーの最小値と最大値を新たに保持するために、オーバーフロー・ブロックが発生することが無くなる。

また、本第１の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムによれば、一つのプライマリー・ブロックに対して、オーバーフロー・ブロックが多数、連結されている場合には、目的のレコードを探すために、オーバーフロー・ブロックの中のレコードを順次読んでいく必要があり、検索時間が余分に掛かる、という問題に対しても、まず、プライマリー・ブロックの

主キーの最小値と最大値を読み、ターゲット・キー値がその範囲内であれば、プライマリー・ブロック中に目的のレコードが存在しうることが判定できる。最大値を超えている場合には、オーバーフロー・ブロックに目的のレコードが存在しうることを示しており、一番目のオーバーフロー・ブロックを読んで、そのブロックの主キー値の最小値と最大値と、ターゲット・キー値とを比較し、範囲内であればそのブロック内に目的のレコードが存在しうるし、最大値を超えていれば、次のオーバーフロー・ブロック以降に、目的のレコードが存在しうることが分かる。

このように、本第１の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムによれば、多数のオーバーフロー・ブロックが連結されている場合には、目的のレコードを探す時間が大幅に短縮可能となる。

【０１５２】

［代替キー・テーブルの再編成］

次に、本第２の発明及び第３の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムの共通事項について説明する。本第２の発明及び第３の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムの共通事項は、代替キー・テーブルの再編成に関するものである。以下、代替キー・テーブルの再編成に関して述べる。代替キー・テーブルの形式は、本発明者が既に提案した「データ格納検索方式」で、図に例示してある。

【０１５３】

［代替キー・テーブルの形式］

この形式に関して、本第２の発明及び第３の発明では、より有効な形式を２つ考案した。本発明者が既に提案した「データ格納検索方式」では、代替キー・ブロックは、そのブロック中に含まれるエントリーの代替キー値の最小値最大値と、そのブロックに連結される代替キー・オーバーフロー・ブロックに含まれるエントリーの代替キー値の最小値と最大値を保持する形式となっている。

【０１５４】

まず、第一の形式は、代替キー・ブロックに関して、その代替キー・ブロックに含まれるエントリーの代替キー値の最小値と最大値の組を一つと、その代替キ

ー・ブロックに連結される代替キー・オーバーフロー・ブロック総てに格納されているエントリーの、代替キー値の最小値と最大値の組を一つ保持し、代替キー・オーバーフロー・ブロックでは、その代替キー・オーバーフロー・ブロックに含まれるエントリーの代替キー値の最小値と最大値を保持する、という形式とするものである。この第一の形式は、第２の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムの対象となるものである。

【０１５５】

このようにすることで、代替キー・ブロックに複数の代替キー・オーバーフロー・ブロックが連結されている場合に、目的のエントリーを検索する際の負荷を軽減できる。代替キー・ブロックに、代替キー・オーバーフロー・ブロックの代替キー値の最小値と最大値を保持し、代替キー・オーバーフロー・ブロックには、代替キー値の最小値と最大値を保持しない場合には、目的のエントリーが、どの代替キー・オーバーフロー・ブロックに格納されているかを、エントリーを順次検索していかなければならなかったが、ここで提示する形式とすることにより、代替キー・オーバーフロー・ブロックの代替キー値の最小値と最大値を読んで、その範囲でなければ、次以降の代替キー・オーバーフロー・ブロックに存在しうるので、検索の負荷が低減する。

【０１５６】

しかしながら、代替キー・ブロックと代替キー・オーバーフロー・ブロックの形式が異なるため、代替キー・オーバーフロー・ブロックを代替キー・ブロックに変換するための負荷が増大する。

次に、代替キー・テーブルの第二の形式は、代替キー・テーブルにもロケーション・テーブルを使用する、というものであるが、この形式に関しては、第一の形式に関する再編成の説明の後に詳述する。この第二の形式は、本第３の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムの対象となるものである。

【０１５７】

[代替キー・テーブルの再編成の目的]

ロケーション・テーブル及びブロックの再編成の目的と同様である。代替キー

・オーバーフロー・ブロックの解消、フラグメンテーションの解消、適正な初期格納率の確保の３つである。何れの目的も、ロケーション・テーブル及びブロックの再編成で詳しく述べてあるので、ここでは、個別に関する詳しい説明は省略する。

【０１５８】

[代替キー・テーブルが第一の形式である場合の再編成]

次に、本第２の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムについて説明する。本第２の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいても、図１と、図９ないし図１０を参照して説明する。

【０１５９】

ここで、図９は、本第２の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいて、再編成を説明するための図である。

図１０は、本第２の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいて、フラグメンテーションを解消するための動作を説明するための図である。

これらの図において、図１に例示されているデータベースに対して、代替キー・テーブルが第一の形式である場合の再編成に関して図９を用いて説明する。図１では、３種類の代替キー（１１Ａ，１１Ｂ，１１Ｃ）が存在しているが、代替キー１１Ａの再編成の場合に関して説明を行う。代替キー１１Ｂ，１１Ｃに対しても、全く同じロジックで実現可能であるので、これらについては説明を省略する。

【０１６０】

この図９では、代替キー・オーバーフロー・ブロックの解消に関しての図解が主となっているが、フラグメンテーションの解消、適正な初期格納効率の確保の３点を同時に実施することは、ロケーション・テーブル及びブロックの再編成時の場合と同様である。

この図９において、代替キー・テーブル１１Ａは、現用代替キー・ブロックＡＡＣと、新規代替キー・ブロックＡＡＮとを備え、かつ、代替キー・オーバーフ

ロー・ブロック１５，１６とからなる。また、図９において、現用代替キー・ブロックＡＡＣの３番目には、代替キー・オーバーフロー・ブロック１５（代替キー・ブロック番号「２－２」）が連結され、この代替キー・オーバーフロー・ブロック１５（代替キー・ブロック番号「２－２」）には代替キー・オーバーフロー・ブロック１６（代替キー・ブロック番号「２－３」）が連結されている。また、図９において、現用代替キー・ブロックＡＡＣの６番目には、代替キー・オーバーフロー・ブロック１５（代替キー・ブロック番号「５－２」）が連結されている。また、図９において、現用代替キー・ブロックＡＡＣの７番目には、代替キー・オーバーフロー・ブロック１５（代替キー・ブロック番号「６－２」）が連結され、この代替キー・オーバーフロー・ブロック１５（代替キー・ブロック番号「６－２」）には代替キー・オーバーフロー・ブロック１６（代替キー・ブロック番号「６－３」）が連結されている。さらに、図９において、現用代替キー・ブロックＡＡＣの１０番目には、代替キー・オーバーフロー・ブロック１５（代替キー・ブロック番号「９－２」）が連結されている。

【０１６１】

＜２つの代替キー・テーブルと再編成ポインター＞

本形式の場合の再編成では、代替キー・テーブル１１Ａでは、現用代替キー・ブロックＡＡＣと現用代替キー・ブロックＡＡＮの２つを、プライマリー・システム１上に設けることにより、再編成を実現する。つまり、図９に示すように、現用代替キー・ブロックＡＡＣに対して、現用代替キー・ブロックＡＡＮを設ける。また、ロケーション・テーブルとブロックの再編成で用いたように、再編成ポインターを使用する。現用代替キー・ブロックＡＡＣ用の再編成ポインターを現再編成ポインターＲＰＡＡＣとし、現用代替キー・ブロックＡＡＮ用の再編成ポインターを新再編成ポインターＲＰＡＡＮとする。

【０１６２】

[代替キー・テーブルが第一の形式である場合の再編成：代替キー・オーバーフロー・ブ ロックの解消]

図９では、２点鎖線が使用されているが、この鎖線は書き移しを主とする移動を表している。

現用代替キー・ブロックＡＡＣの大きさは、図９からもわかるように、現用代替キー・ブロックＡＡＣの代替キー・ブロックと代替キー・オーバーフロー・ブロックの数を足したものに、代替キー・ブロックのサイズを掛けた積と等しくなるが、第１の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいてロケーション・テーブルとブロックの再編成で説明したように、再編成中に代替キー・オーバーフロー・ブロックが発生して、ブロック数が増加するので、多めに確保する。

【０１６３】

しかしながら、他方で、フラグメンテーションを解消することにより、必要ブロック数が少なくなる場合もあるし、適正な初期格納率の確保によっても、必要ブロック数は変動するので、格納されているエントリーの容量と適正な初期格納率を元に計算するのが、最も好ましい方法である。

まず、現用代替キー・ブロックＡＡＮの領域の確保を連続領域内に行う。現再編成ポインターＲＰＡＡＣの初期値は、現用代替キー・ブロックＡＡＣの先頭アドレスである。新再編成ポインターＲＰＡＡＮの初期値は、現用代替キー・ブロックＡＡＮの先頭アドレスである。

【０１６４】

＜代替キー・ブロックの書き移し＞

まず、現用代替キー・ブロックＡＡＣの１番目の代替キー・ブロック（代替キー・ブロック番号「０」）を排他する。

次に、現用代替キー・ブロックＡＡＣの1番目の代替キー・ブロック（代替キー・ブロック番号「０」）を現用代替キー・ブロックＡＡＮの1番目の代替キー・ブロックに書き移す（図９のＳ４１）。このブロックには、代替キー・オーバーフロー・ブロックが連結されていないので、現再編成ポインターＲＰＡＡＣの内容を、現用代替キー・ブロックＡＡＣの２番目のブロック（代替キー・ブロック番号「１」）のアドレスに書き換え、新再編成ポインターＲＰＡＡＮの内容を、現用代替キー・ブロックＡＡＮの２番目のブロック（代替キー・ブロック番号「１」）のアドレスに書き換える。

【０１６５】

ここでは、現用代替キー・ブロックＡＡＣのm番目の代替キー・ブロックを、現用代替キー・ブロックＡＡＮのｎ番目の代替キー・ブロックに書き移す、という表現を用いているが、実際のロジックとしては、現再編成ポインターＲＰＡＡＣが指している現用代替キー・ブロックＡＡＣの代替キー・ブロックを、新再編成ポインターＲＰＡＡＮが指している現用代替キー・ブロックＡＡＮの代替キー・ブロックに書き移す、ということになる。

次に、現用代替キー・ブロックＡＡＣの１番目の代替キー・ブロック（代替キー・ブロック番号「０」）の排他を解除する。

次に、現用代替キー・ブロックＡＡＣの２番目の代替キー・ブロック（代替キー・ブロック番号「１」）を排他する。

次に、現用代替キー・ブロックＡＡＣの２番目のブロックを、現用代替キー・ブロックＡＡＮの２番目のブロックに書き移す（図９のＳ４２）。書き移し後、現再編成ポインターＲＰＡＡＣと新再編成ポインターＲＰＡＡＮの指しているアドレスを次の代替キー・ブロックのアドレスに書き換える。

【０１６６】

次に、現用代替キー・ブロックＡＡＣの２番目の代替キー・ブロック（代替キー・ブロック番号「１」）の排他を解除する。

さらに、現用代替キー・ブロックＡＡＣの３番目のブロックの再編成であるが、この代替キー・ブロックには、２つの代替キー・オーバーフロー・ブロック１５，１６が連結されている。この場合、排他は、現用代替キー・ブロックＡＡＣの代替キー・ブロック番号「２」と、それに連結されている、代替キー・オーバーフロー・ブロック１５（代替キー・ブロック番号「２－２」）と、代替キー・オーバーフロー・ブロック１６（代替キー・ブロック番号「２－３」）に対して掛けておく。

【０１６７】

その後、まず、現用代替キー・ブロックＡＡＣの代替キー・ブロック番号「２」を、現用代替キー・ブロックＡＡＮの代替キー・ブロック番号「２」に書き移す（図９のＳ４２）。新再編成ポインターＲＰＡＡＮ０を現用代替キー・ブロックＡＡＮの代替キー・ブロック番号「３」の先頭を指すように書き換える。

次に、現用代替キー・ブロックＡＡＣの代替キー・ブロック番号「２」に代替キー・オーバーフロー・ブロック１５（代替キー・ブロック番号「２－２」）が連結されているので、代替キー・オーバーフロー・ブロック１５（代替キー・ブロック番号「２－２」）を新再編成ポインターＲＰＡＡＮが指しているブロック（現用代替キー・ブロックＡＡＮの代替キー・ブロック番号「３」）に書き移す（図９のＳ４３）。新再編成ポインターＲＰＡＡＮを現用代替キー・ブロックＡＡＮの代替キー・ブロック番号「４」の先頭を指すように書き換える。

【０１６８】

次に、代替キー・オーバーフロー・ブロック１５（代替キー・ブロック番号「２－２」）に代替キー・オーバーフロー・ブロック１６（代替キー・ブロック番号「２－３」）が連結されているので、代替キー・オーバーフロー・ブロック１６（代替キー・ブロック番号「２－３」）を新再編成ポインターＲＰＡＡＮが指しているブロック（現用代替キー・ブロックＡＡＮの代替キー・ブロック番号「４」）に書き移す（図９のＳ４４）。ついで、新再編成ポインターＲＰＡＡＮを現用代替キー・ブロックＡＡＮの代替キー・ブロック番号「５」の先頭を指すように書き換える。

【０１６９】

ここで、現用代替キー・ブロックＡＡＣの代替キー・ブロック番号「２」と代替キー・オーバーフロー・ブロック１５（代替キー・ブロック番号「２－２」）と、代替キー・オーバーフロー・ブロック１６（代替キー・ブロック番号「２－３」）の排他を解除する。

次に、現用代替キー・ブロックＡＡＣの代替キー・ブロック番号「３」のブロックを、現用代替キー・ブロックＡＡＮの代替キー・ブロック番号「５」に書き移す（図９のＳ４５）。この際にも、該当するブロックを排他しておく。書き移し後、現再編成ポインターＲＰＡＡＣと新再編成ポインターＲＰＡＡＮの指しているアドレスを次の代替キー・ブロックのアドレスに書き換える。以上の動作が完了すると、図８に示す状態になる。

【０１７０】

［代替キー・テーブルが第一の形式である場合の再編成：オーバーフローの解

消の例外]

以上の説明では、代替キー・オーバーフロー・ブロックの解消に関して述べたが、解消ができない場合について説明する。これは、同一代替キーを保有するエントリーが多くて、一つのブロックに格納できない場合である。同一の代替キーを持つエントリーは、複数の代替キー・ブロックに格納されることは無く、必ず一つの代替キー・ブロックと１つ以上の代替キー・オーバーフロー・ブロック、または、複数の代替キー・オーバーフロー・ブロックに格納される。複数の代替キー・オーバーフロー・ブロックに格納されるとは、代替キー・ブロックとその直後の代替キー・オーバーフロー・ブロックには既にエントリーが格納されており、代替キー・オーバーフロー・ブロックの途中から同一代替キーのエントリーの格納が開始されるような場合である。

【０１７１】

同一代替キーのエントリーが格納されている場合は、代替キー・ブロック及び代替キー・オーバーフロー・ブロックにその情報を保持するのが分かりやすい。同一代替キーのエントリーの先頭部分か、中間部分か、最後尾部分か、といった情報を保持するのである。それ以外は、通常のエントリーを格納する場合と同様である。

この場合の、代替キー・オーバーフロー・ブロックの解消は、通常のエントリーの部分に関しては、解消を行えるが、同一代替キーのエントリーが格納される代替キー・ブロック及び代替キー・オーバーフロー・ブロックに関しては、必ず代替キー・オーバーフロー・ブロックを伴った形式となるため、代替キー・オーバーフロー・ブロックの解消はできない。このような場合には、再編成後の情報として、同一代替キーのエントリーに関するものを出力することが好ましい。

【０１７２】

[代替キー・テーブルが第一の形式である場合の再編成：フラグメンテーションの解消]

上記では、代替キー・オーバーフロー・ブロックを解消する方法に関して述べたが、オーバーフローと同様にフラグメンテーションも、効率上では大きな問題であった。

これを解消するためには、代替キー・オーバーフロー・ブロックを解消する再編成と似た動作で実現できることを、図１０を用いて説明する。図１０において、現用代替キー・ブロックＡＡＣのブロック番号「０」、「１」、代替キー・オーバーフロー・ブロック１５（ブロック番号「１－２」）、代替キー・オーバーフロー・ブロック１６（ブロック番号「１－３」）、現用代替キー・ブロックＡＡＣのブロック番号「２」、代替キー・オーバーフロー・ブロック１５（ブロック番号「２－２」）には、適度にレコードが格納されており、代替キー・オーバーフロー・ブロックの解消を前述の方法で完了したとする。以下の説明で使用する代替キー・ブロック番号は、現用代替キー・ブロックＡＡＣに基づく番号を使用する。現用代替キー・ブロックＡＡＮに基づく番号を使用する場合には、その旨を記述する。

【０１７３】

現用代替キー・ブロックＡＡＣの代替キー・ブロック番号「３」には、格納容量の３０％のエントリーが格納されている。現用代替キー・ブロックＡＡＣの代替キー・ブロック番号「４」には格納容量の４０％のエントリーが格納されている。現用代替キー・ブロックＡＡＣの代替キー・ブロック番号「５」には７０％、代替キー・オーバーフロー・ブロック１５のブロック番号「５－２」には６０％、現用代替キー・ブロックＡＡＣの代替キー・ブロック番号「６」には７０％のエントリーが格納されている。また、再編成後の各代替キー・ブロックの適正な初期格納率は９０％とする。これは、再編成後に挿入エントリーが発生した場合に、その都度代替キー・オーバーフロー・ブロックを発生させないために設けることができる。

【０１７４】

本第２の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムは、現用代替キー・ブロックＡＡＣの現再編成ポインターＲＰＡＡＣが指している代替キー・ブロック番号「３」の再編成を行おうとする。しかしながら、格納率（代替キー・ブロックの容量に対する、その代替キー・ブロックに格納されているエントリーの容量の割合）が３０％であるため、適正な初期格納率を満たしていない。このため、現用代替キー・ブロックＡＡＣの次の代替キー・

ブロック番号「４」を見る。これも格納率が４０％であるため、２つの代替キー・ブロックを足しても適正な初期格納率（９０％）を下回ることになる。更に代替キー・ブロック５を見ると格納率は７０％であるため、適正な初期格納率９０％を上回ることになる。現用代替キー・ブロックＡＡＣの代替キー・ブロック番号「３」に格納されているエントリーはそのまま残し、現用代替キー・ブロックＡＡＣの代替キー・ブロック番号「４」に入っていたエントリーを、現用代替キー・ブロックＡＡＣの代替キー・ブロック番号「３」にする。これにより、現用代替キー・ブロックＡＡＣの代替キー・ブロック番号「３」は格納率が７０％となる。適正な初期格納率にするため、現用代替キー・ブロックＡＡＣの代替キー・ブロック番号「５」に格納されているエントリーの先頭２０％を、現用代替キー・ブロックＡＡＣの代替キー・ブロック番号「３」に移動すると同時に、残りのエントリー５０％を前方（図示左側）に詰める。

【０１７５】

現用代替キー・ブロックＡＡＣの代替キー・ブロック番号「３」における格納率が適正になったので、現用代替キー・ブロックＡＡＣの代替キー・ブロック番号「３」を、現用代替キー・ブロックＡＡＮの代替キー・ブロック番号「６」に書き移す。新再編成ポインターＲＰＡＡＮを現用代替キー・ブロックＡＡＮの代替キー・ブロック番号「７」の先頭に移動させる。現用代替キー・ブロックＡＡＣの代替キー・ブロック番号「４」は使用されない代替キー・ブロックであり書き移しされない。

【０１７６】

次に、現用代替キー・ブロックＡＡＣの代替キー・ブロック番号「５」を操作するのであるが、現用代替キー・ブロックＡＡＣの代替キー・ブロック番号「３」，「４」の再編成により、現用代替キー・ブロックＡＡＣの代替キー・ブロック番号「５」は格納率が５０％となっている。代替キー・オーバーフロー・ブロック１５（ブロック番号「５－２」）は格納率が６０％であるので、代替キー・オーバーフロー・ブロック１５（ブロック番号「５－２」）のエントリーの内、先頭から３０％のエントリーを、現用代替キー・ブロックＡＡＣの代替キー・ブロック番号「５」に移動し、同時に代替キー・オーバーフロー・ブロック１５（

ブロック番号「５－２」）に残っているエントリーを前方（図示左側）に詰める。更に、現用代替キー・ブロックＡＡＣの代替キー・ブロック番号「５」が代替キー・オーバーフロー・ブロック１５（ブロック番号「５－２」）を代替キー・オーバーフロー・ブロックとして連結しているのを切断する。これは、現用代替キー・ブロックＡＡＣの代替キー・ブロック番号「５」の代替キー・オーバーフロー・ブロック・アドレスを特定の値（例えばゼロ）にセットする。

【０１７７】

現用代替キー・ブロックＡＡＣの代替キー・ブロック番号「５」の格納率が適正なものになったので、現用代替キー・ブロックＡＡＣの代替キー・ブロック番号「５」を新再編成ポインターＲＰＡＡＮが差している現用代替キー・ブロックＡＡＮの代替キー・ブロック番号「７」に書き移す。新再編成ポインターＲＰＡＡＮを現用代替キー・ブロックＡＡＮの代替キー・ブロック番号「８」の先頭に移動する。

【０１７８】

次に、代替キー・オーバーフロー・ブロック１５（ブロック番号「５－２」）の格納率が３０％であるので、現用代替キー・ブロックＡＡＣの次の代替キー・ブロック番号「６」を見ると格納率が６０％であるので、現用代替キー・ブロックＡＡＣの代替キー・ブロック番号「６」のエントリー総てを代替キー・オーバーフロー・ブロック１５（ブロック番号「５－２」）に移動する。代替キー・オーバーフロー・ブロック１５（ブロック番号「５－２」）の格納率が適正になったところで、現用代替キー・ブロックＡＡＣの代替キー・オーバーフロー・ブロック１５（ブロック番号「５－２」）を現用代替キー・ブロックＡＡＮの代替キー・ブロック番号「８」に書き移す。新再編成ポインターＲＰＡＡＮを現用代替キー・ブロックＡＡＮの代替キー・ブロック番号「９」の先頭に移動する。この時点で、現用代替キー・ブロックＡＡＣの代替キー・ブロック番号「６」は使用されない代替キー・ブロックとなり書き移しされない。

このように次々と処理されてゆき、現用代替キー・ブロックＡＡＣの代替キー・ブロック番号「８」まで再編成が完了した時点を、図１０に示している。

【０１７９】

［代替キー・テーブルが第一の形式である場合の再編成：適正な初期格納率の確保］

また、フラグメンテーションとは逆に、適正な初期格納率を確保するために、代替キー・ブロックの追加が必要な場合も発生する。これは、例えば、すべての代替キー・ブロックの格納率が１００％であった場合に、適正な初期格納率を９０％とすると、一時点で９個の代替キー・ブロックを再編成の対象にし、１つの代替キー・ブロックを追加して１０個の代替キー・ブロックに対して、エントリーを９０％ずつ格納する、というような形態である。

適正な初期格納率の確保を厳密に行おうとする、代替キー・ブロックとエントリーの大きさの関係で不可能な場合や、一時点でかなり多数の代替キー・ブロックを再編成の対象にしなければならなくなる可能性があるため、排他範囲が広がり、システムの稼動に悪影響を与える可能性がある。

【０１８０】

＜複数の初期格納率＞

このような状態が発生することを防止する上で、適正な初期格納率を、例えば、８５％から９０％など、複数の値を用いることが、運用上は好ましい。この場合、適正な初期格納率が、８５％から９０％の範囲に入っていればよい、とするものである。

この適正な初期格納率の確保を行う場合には、代替キー・ブロックの書き移しを単純に行うのではなく、代替キー・ブロック内でエントリーの再編成を行いながら、代替キー・ブロックの書き移しを行うことになる。

【０１８１】

［代替キー・テーブルが第一の形式である場合の再編成：実際の再編成とデッドロックの防止］

実際に再編成を行う場合には、代替キー・オーバーフロー・ブロックの解消と、フラグメンテーションの解消、及び、適正な初期格納率の確保を同時に実行することになるので、上記の３つの方式を組み合わせて使用する。

原理的には以上のような方法で実現が可能であるが、代替キー・ブロックを順次読んでいき、適正な初期格納率の代替キー・ブロックを作成していくと、当初

から排他の範囲が確定せず、順次に排他していくために、デッドロックが発生する可能性が増加する。デッドロックの発生を防ぐためには、次のような方法が有効である。

【０１８２】

これは、ロケーション・テーブルとブロックの再編成で述べたのと同様の方法である。まず、排他を掛けずに次々と代替キー・ブロックを読みだし、エントリーの格納率を調べて、複数の代替キー・ブロックを組み合わせて、適正な大きさとなるように計算をした上で、排他範囲を決定する方法である。

このようにして、代替キー・ブロック内の格納率を調べた後で、その時点での再編成範囲を確定し、その範囲の代替キー・ブロック及び、代替キー・ブロックに連結されている代替キー・オーバーフロー・ブロックの排他を行う。その後、再編成対象となる代替キー・ブロックと代替キー・オーバーフロー・ブロックの数と格納可能容量を調べ、格納対象エントリーの実際の容量を調べた上で、必要になる代替キー・ブロックが現用代替キー・ブロックと代替キー・オーバーフロー・ブロックの和と同数か、増加するか、減少するか、を調べる。この後、代替キー・オーバーフロー・ブロックの解消や、フラグメンテーションの解消、適正な初期格納率の確保で説明したロジックを使用して、代替キー・ブロックの再編成を行う。

この場合、再編成ポインターは、再編成の対象となったブロックの分だけ、一度に移動することになる。

このように、一時点で、１個から数十個の代替キー・ブロックを対象として再編成を実施するのであるが、この再編成は、見かけ上、通常のデータ処理のトランザクションとして扱うことにより、通常のデータ処理との矛盾を発生させない。また、この再編成を次々と、代替キー・ブロックに対して実施することにより、代替キー・ブロック全体の再編成を完了する。

【０１８３】

[代替キー・テーブルが第一の形式である場合の再編成：再編成の完了]

本発明者が既に提案した「データ格納検索方式」では、代替キー・テーブルにおけるラスト・ポインターの記述が無いが、本第２の発明の実施の形態に係るデ

ータ及びデータベースの無停止自動再編成システムでは、代替キー・テーブルに対して代替キー・ラスト・ポインターを使用する方法を追加し、代替キー・ラスト・ポインターを使用することが好ましい。

【０１８４】

代替キー・ラスト・ポインターは、ロケーション・テーブルのラスト・ポインターと同等の機能を有するもので、代替キー・テーブルの代替キー・ブロックを、予め、必要数よりも多くの代替キー・ブロックを設定した場合に、代替キー・テーブルの後方に使用されない代替キー・ブロックが発生する可能性がある。それらの使用されていない代替キー・ブロックがある場合に、そのままバイナリー・サーチを適用すると、目的とするブロックが検索できなくなるので、使用していないブロックを対象外とするための操作が必要であった。

【０１８５】

代替キーは、その性格上、エントリーは挿入になる場合が殆どで、追加になる場合は少ないので、代替キー・テーブルが一旦作成された後では、使用していないブロックを対象外とする操作は必要なくなる場合が殆どであるが、代替キー・ラスト・ポインターを設けることにより、代替キー・テーブルを、どこまで使用しているかが簡単に分かるようになる。このように、代替キー・ラスト・ポインターを用いると、未使用ブロックをバイナリー・サーチの対象外とする操作が不要となる他、再編成完了の判別が簡単になる。

【０１８６】

代替キー・ラスト・ポインターを使用した場合の、代替キー・テーブルの再編成完了を認識する方法に関して説明する。図９又は図１０において、現用代替キー・ブロックＡＡＣには、その代替キー・テーブルを使用している最終位置を示すための、代替キー・ラスト・ポインターが設けてある。代替キー・ラスト・ポインターを設ける目的は、次のようなものである。代替キー・テーブルは連続領域に確保することになっている。代替キー・テーブルの代替キー・ブロックが不足した場合には、最初のロケーション・テーブルとは連続しない領域に追加の代替キー・テーブルを設けて、それらを連続した領域であるかのようにバイナリー・サーチできるよう、アドレス変換する方法が、本発明者が既に提案した「デー

タ格納検索方式」では示されているが、不連続領域が多くなると、アドレス変換の負荷が増大するため、好ましくない。このため、代替キー・テーブル用の領域は、予め十分に大きな領域を確保しておき、使用している代替キー・ブロックと、未使用の代替キー・ブロックを区分するために、使用している代替キー・ブロックの次の代替キー・ブロックのアドレスを指す、ということになっている。

このように、代替キー・ラスト・ポインターを設けることにより、現用代替キー・ブロックＡＡＣの先頭アドレスと代替キー・ラスト・ポインターとの間の代替キー・ブロックに対してバイナリー・サーチを実行することにより、現用代替キー・ブロックＡＡＣに未使用代替キー・ブロックが存在していても、代替キーによるエントリー検索が可能となる。

【０１８７】

＜再編成の完了の検出方法＞

この代替キー・ラスト・ポインターを再編成完了の指標として用いる。再編成を実行して行き、現再編成ポインターＲＰＡＡＣが指すアドレスが、代替キー・ラスト・ポインターが指すアドレスと一致した場合に、代替キー・テーブルの再編成が完了したことになる。尚、最終代替キー・ブロックに、代替キー・オーバーフロー・ブロックが連結されている場合でも、それらの、代替キー・オーバーフロー・ブロックの再編成が完了しない限り、ＲＰＡＡＣの移動は行われないので、問題ない。

【０１８８】

代替キー・ラスト・ポインターを使用しない場合の再編成完了の判別は、現用代替キー・ブロックＡＡＣを順次再編成していき、使用されていない代替キー・ブロックがある場所で完了する。しかしながら、この場合には、当初、代替キー・ブロックに格納されていたエントリーが、すべて削除されてしまった場合に、当該、代替キー・ブロックが未使用であるかのように判定される可能性があるため、エントリーが削除されてしまったブロックは、次のように、特殊なフォーマットを採用することが好ましい。

代替キー・ブロックの、代替キー値の最小値と最大値は、その代替キー・ブロックが空になる直前に格納されていたエントリーの代替キー値を、そのまま保持

する。そして、空のブロックであることを示すために、例えば、代替キー・ブロック内のキー値以外の領域を「ヌル（ＮＵＬＬ）」で満たす。「ヌル」で満たす他に、空であるか否かのフラグを設けて、設定するようにしてもよい。

【０１８９】

［代替キー・テーブルが第一の形式である場合の再編成：未使用ブロックの再利用］

ロケーション・テーブルとブロックの再編成を行う場合には、ブロック自体の書き移しは行わないので、未使用ブロックの再利用が問題となったが、上記で説明した代替キー・テーブルの再編成では、現用代替キー・テーブルから、現用代替キー・テーブルへ、代替キー・ブロックを書き移すことによって実現しているので、未使用ブロックの問題点は基本的には発生しない。

しかしながら、代替キー・オーバーフロー・ブロックの取得が、全体の格納領域で散在した状態で行われている場合には、その代替キー・オーバーフロー・ブロックが占めていた領域が使用しにくくなるため、未使用領域になってしまう可能性は存在する。

これに対処するためには、ロケーション・テーブルとブロックの再編成で述べたように、未使用ブロック・アロケーション・テーブルを作成し、未使用となった元代替キー・オーバーフロー・ブロックを管理し、現用代替キー・テーブルを使用し始めた後に発生する代替キー・オーバーフロー・ブロックには、元代替キー・オーバーフロー・ブロックが占めていた格納領域を割り当てることで、領域利用の有効化を図ることが可能となる。

【０１９０】

［代替キー・テーブルが第一の形式である場合の再編成：再編成中のデータベースのアクセス］

次に、本第２の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいて、再編成中の、データの検索・読み書きに関して、再編成を行っているときにも、データの検索・読み書きが可能であることを述べる。付言すれば、再編成中にデータの検索・読み書きが可能であるということが、「データ格納・検索システム」の稼動を行ったまま、つまり、システムを停止する

ことなく、再編成を実施できるということになる。

【０１９１】

再編成を行っていない状態での、代替キー値による検索では、現用代替キー・ブロックＡＡＣを用いてバイナリー・サーチを行う。再編成を実施中である場合には、ターゲット・キー値（検索の対象となるキー値）が再編成ポインターＲＰＡＡＣの代替キー値より小さい（図示上方にある）か、大きいか（図示下方にある）かの判定を行う。これは、現用代替キー・ブロックＡＡＣが代替キー値の順に並んでいることから、現再編成ポインターＲＰＡＡＣの示している代替キー・ブロックのキー値とターゲット・キー値の比較を行えばよい。

ターゲット・キー値が、新再編成ポインターＲＰＡＡＮが指している代替キー・ブロックの代替キー値の最小値より小さい場合は、新再編成ポインターＲＰＡＡＮより上方（アドレスが小さい）に、目的とするエントリーが存在することになる。この場合は、現用代替キー・ブロックＡＡＮを使用して、現用代替キー・ブロックＡＡＮの先頭アドレスと新再編成ポインターＲＰＡＡＮとの間でバイナリー・サーチを行う。バイナリー・サーチの結果、目的のエントリーが指している代替キー・ブロックの中のエントリーを調べて、目的のレコードが存在するか否かを検出する。

【０１９２】

ターゲット・キー値が現再編成ポインターＲＰＡＡＣの指している代替キー・ブロックの代替キー値の最小値と等しいか大きい場合には、目的とするエントリーは、現再編成ポインターＲＰＡＡＣより下方（ＲＰＡＡＣ０が指しているエントリー、または、アドレスが大きい）に存在することになる。この場合は、現用代替キー・ブロックＡＡＣを使用して、現再編成ポインターＲＰＡＡＣと現用代替キー・ブロックＡＡＣの代替キー・ラスト・ポインターとの間のエントリーに対してバイナリー・サーチを行う。

【０１９３】

このように、目的の代替キー・ブロック及びエントリーの検索が確実に実行できるので、目的とする代替キー値を持つエントリーの検索が可能となる。

代替キーのエントリーの検索ができた後は、その代替キーのエントリーが、プ

ロック番号を含んでいる場合は、ブロック番号を元に、ロケーション・テーブルのエントリーを見つけ、そのエントリーが指しているブロックから、目的のレコードを検索する。代替キーのエントリーがブロック・アドレスを含んでいる場合は、ブロックの位置が分かるので、直ちにブロック内のレコードの検索を行う。代替キーのエントリーが、ブロック番号も、ブロック・アドレスも保持していない場合は、そのエントリーの主キー値によって、ロケーション・テーブルの検索を行い、ロケーション・テーブルのエントリーが指しているブロック内をサーチして、目的のレコードを検索する。

【０１９４】

上記の説明では、代替キー・テーブルを再編成中である場合に関して述べた。複数のテーブルを同時に再編成しないので、代替キー・テーブルの再編成中である場合には、ロケーション・テーブルは再編成中では無い、ということになる。これにより、ロケーション・テーブルとブロックの再編成で述べたような、現用と新規のロケーション・テーブルが２つ存在する、ということはないので、何れを使用するかの工夫は不要である。

【０１９５】

また、目的とする代替キー値を持つエントリーが含まれる代替キー・ブロックが、正に再編成中で、排他されている場合には排他待ちとなり、その排他が解除されるまでは、当該ブロックのアクセスはできないことになるが、これは、通常のアクセスでレコードの更新や挿入、削除が行われる場合と同様である。言い換えれば、排他中のブロックに対しての要求は、排他待ちになるが、そのブロックに対する再編成が完了して、排他が解除されれば、処理が可能となる。

上記では、レコード検索の場合に関して述べたが、これを応用することで、代替キーによる、レコードの更新、削除を行うことも可能である。レコードの挿入は、主キーで実行するので、ここでは対象外となる。また、レコードの削除は代替キーがノン・ユニークであるため、実行する場合には注意が必要である。

【０１９６】

代替キーによる、レコードの更新・削除は、主キーを用いてロケーション・テーブルを検索するのに代えて、最初の検索を代替キー・テーブルを使用して行う

点が主キーの場合と異なるが、代替キー・エントリーを見つけた後は、代替キー・エントリーに含まれるブロック番号等の情報と主キー値により、ロケーション・テーブルを検索して、レコードが格納されているブロックを見つけ、ブロック内で目的のレコードを見つけるという順序となる。レコードを見つけた後の、更新や削除、それに伴う代替キー値の変更に関しては、主キー値で更新、削除を行う場合と全く同様である。しかしながら、この場合には、次に述べるような可能性がある。

【０１９７】

[代替キー・テーブルが第一の形式である場合の再編成：検索中に再編成が進行した場合への対処]

このように再編成ポインターを使用し、ターゲット・キー値と比較することで、現用代替キー・ブロックＡＡＣ、現用代替キー・ブロックＡＡＮの何れかを使用するかを決定することにより、再編成中でもレコードの呼び出しが可能であることを説明した。

【０１９８】

次に、本第２の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいて、代替キー・テーブルが第一の形式である場合に、検索中に再編成が進行した場合への対処について図１１を参照して説明する。

ここに、図１１は、本第２の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいて、検索中に再編成が進行した場合への対処について説明するための図である。

ここで、図１１に示すように、現用代替キー・ブロックＡＡＣに対して代替キーの検索を行っている最中に再編成が進んでしまい、検索開始時の現用再編成ポインターＲＰＡＡＣと検索終了時の現用再編成ポインターＲＰＡＡＣの位置が異なっていて、その範囲の代替キー・ブロックが検索対象となった場合には、第１の発明の実施の形態で説明したようなロケーション・テーブルとブロックの再編成のような問題は発生しない。

【０１９９】

第１の発明の実施の形態におけるロケーション・テーブルとブロックの再編成

時には、オーバーフロー・ブロックの連結が切り離されるため、ロケーション・テーブルのエントリーからブロックを見にいった場合に、再編成前のオーバーフロー・ブロックにあったレコードが検出できなくなってしまう場合があったため、工夫が必要であった。

しかしながら、図１１に示すように代替キー・テーブル１１Ａが第一の形式である場合のような本第２の発明の実施の形態においては、再編成は、元の代替キー・ブロックＡＡＣ及び、代替キー・オーバーフロー・ブロック１５，１６を残したまま、新しい代替キー・テーブルＡＡＮを作成する。また、代替キー・ブロックＡＡＣ及び、代替キー・オーバーフロー・ブロック１５，１６中のエントリーは、再編成によって影響を受けない。

このため、現用代替キー・テーブル１１Ａ（の代替キー・ブロックＡＡＣ）をバイナリー・サーチで検索している間に、再編成が進み、図１１に示すように再編成ポインターＲＰＡＡＣの位置がずれていても、現用代替キー・テーブル１１Ａ（の代替キー・ブロックＡＡＣ）は再編成による変更が無いため、そのまま使用できる。

【０２００】

[再編成の中断と再開]

以上で説明したように、代替キー・テーブルの再編成中であっても、代替キー値による、レコードの検索（読み込み）や、更新、レコードの挿入、追加、削除が可能であることを説明した。つまり、再編成がどの時点、言い換えれば、代替キー・テーブルのどのブロックまで進行していても、レコードのアクセスが可能であることはいうまでもない。

【０２０１】

＜再編成の中断と再開＞

これに基づけば、一時的に再編成を中断しても、レコードのアクセスは問題なく実行できることが分かる。勿論、中断は、あるまとまりの代替キー・ブロックに対する再編成が終了した後に行われることは当然である。

本第２の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいて、再開は、中断した時点で、現用の再編成ポインターＲＰＡＡ

Ｃと新規の再編成ポインターＲＰＡＡＮが示している、現用の代替キー・テーブルＡＡＣと新規の代替キー・テーブルＡＡＮのエントリーから再編成を実施すればよい。

この機能により、プライマリー・システムの負荷が増加した場合には、再編成を中断して、データ処理に能力を振り分け、データ処理の負荷が減少してきたら、再編成を再開することが可能となるので、再編成の実施に当たって、プライマリー・システムの負荷や運転状況に関して事前予測を行い、一定の時間を予め確保する必要がない。

【０２０２】

代替キー・テーブルのエントリーに関しては、次の３種類が存在することを述べた。すなわち、ブロックの番号を保持する、ブロックのアドレスを保持する、ブロックの番号、アドレスの何れも保持しない、の３種類である。また、これらについては、以下の特徴があることも述べた。

ブロックの番号、アドレスの何れも保持しない場合には、再編成に要する負荷や時間が短縮できるが、代替キーを用いた検索では代替キー・エントリーを検索後にロケーション･テーブルの検索が必要になるため、検索の負荷が増加する。

一方で、ブロックの番号を保持する、ブロックのアドレスを保持する場合には、再編成時には番号やアドレスが変更になる場合に、情報の変更が必要になり、再編成の負荷が大きくなるが、一方で、代替キーを使用した検索は効率的となる。

ここで、代替キー・エントリーの形式を、何れかに決定して代替キー・テーブルを作成して運用をしている場合、当初の予定と状況が変わってしまい、例えば、再編成が頻繁に必要になったとか、逆に、当初予定していた頻度に比較すると、再編成の頻度が著しく低下した場合、などが考えられる。

【０２０３】

このような場合、代替キー・テーブルの再編成時に、エントリーの形式を変更して、検索や追加・更新・削除等の効率は低下させても再編成の速度を向上させるようにする、または逆に、再編成の速度を低下させても検索や追加・更新・削除等の効率を高める、といった切り替えが出来ると好都合である。

これは、次のように代替キー･テーブルを再編成する場合に、代替キー・エントリーの形式を変更することにより可能となる。

ブロック番号やブロック・アドレスを保持しない形式の代替キー・エントリーに対して、ブロック番号やアドレスを付加する場合には、１代替キー・ブロック当たり、代替キー・ブロックの中に既に存在する代替キー・エントリーの数と、ブロック番号またはブロック・アドレスの分だけ情報量が増加することになる。

一時点で、１個または複数個の代替キー・ブロックを対象にして、オーバーフローの解消、フラグメンテーションの解消、適正な初期格納率の確保を行う場合に、情報の増加分が書き込めるように代替キー・ブロックの確保を行う。対象となった代替キー・ブロックのエントリーに関しては、ブロック番号またはブロック・アドレスを追加し、エントリー情報の書き換えを行う。

【０２０４】

この場合、対象となる代替キー・エントリー一つづつについて、そのエントリーの主キー値に基づいて、ロケーション･テーブルの検索を行い、見つかったブロックの番号、またはブロック・アドレスを新規の代替キー・テーブルのエントリーに付加する。

逆に、ブロック番号またはブロック・アドレスを削除する場合には、１代替キー・ブロック当たり、代替キー・ブロックの中に既に存在する代替キー・エントリーの数と、ブロック番号またはブロック・アドレスの分だけ情報量が減少することになる。

一時点で、１個または複数個の代替キー・ブロックを対象にして、オーバーフローの解消、フラグメンテーションの解消、適正な初期格納率の確保を行う場合に、情報の減少分を考慮して代替キー・ブロックの確保を行う。対象となった代替キー・ブロックのエントリーに関しては、ブロック番号またはブロック・アドレス削除し、エントリー情報の書き換えを行う。

付加する場合に比較すると、情報を削除するだけでよいので、時間的には短くて済む。

【０２０５】

[代替キー・テーブルの第二の形式]

＜代替キー・ロケーション・テーブルの追加＞

次に、本第３の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムについて図１２以降の図面を参照して説明する。

【０２０６】

図１２は第３の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムを説明するための図である。

この図１２において、本発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムは、代替キー・テーブル１１Ａ，１１Ｂ，１１Ｃに第二の形式を採用したものであり、代替キー・テーブル１１Ａ，１１Ｂ，１１Ｃにもロケーション・テーブルを使用する、というものである。

【０２０７】

すなわち、本第３の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムでは、代替キー・テーブル１１Ａ，１１Ｂ，１１Ｃの代替キー・ブロック１７を管理する目的で使用するものを、代替キー・ロケーション・テーブルＡＡＬＣと呼ぶことにする。この代替キー・ロケーション・テーブルＡＡＬＣは、本第１の発明の実施の形態において説明したブロックに対するロケーション・テーブルと同様の機能を持つものである。

【０２０８】

図１１において、代替キー・テーブル１１Ａは代替キー・ロケーション・テーブルＡＡＬＣと代替キー・ブロック１７とからなり、代替キー・テーブル１１Ｂも代替キー・ロケーション・テーブルＡＡＬＣと代替キー・ブロック１７とからなり、さらに、代替キー・テーブル１１Ｃも代替キー・ロケーション・テーブルＡＡＬＣと代替キー・ブロック１７とからなる。

したがって、本第３の実施の形態に係るデータ及びデータベースの無停止自動再編成システムでも、代替キー・テーブル１１Ａを代表させて説明し、他の代替キー・テーブル１１Ｂ，１１Ｃの説明を省略する。

【０２０９】

図１３は、本第３の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいて、プライマリー・システムの代替キー・テーブルの

一つを説明するために示す図である。

この図１３において、プライマリー・システム１の代替キー・ロケーション・テーブルＡＡＬＣのエントリーは、そのエントリーが管理する代替キー・ブロックＡＡＣのアドレスを保持する。その他に、必要に応じて、その代替キー・ブロックＡＡＣの番号、その代替キー・ブロックＡＡＣ及び、その代替キー・ブロックＡＡＣの連結されている代替キー・オーバーフロー・ブロック１５，１６に格納されている代替キー・エントリーの代替キー値の最小値、最大値の双方または一方を保持する。代替キー・オーバーフロー・ブロック１５，１６は代替キー・ブロックＡＡＣによって管理されるので、代替キー・ロケーション・テーブルＡＡＬＣでは管理しない。

【０２１０】

代替キー・ブロックＡＡＣと代替キー・オーバーフロー・ブロック１５．１６に、当該ブロックに格納されているエントリーの代替キー値の最小値、最大値を保持するのは、第２の発明における第一の形式と同様である。このようにすることで、代替キー・ブロックＡＡＣに複数の代替キー・オーバーフロー・ブロック１５，１６が連結されている場合に、目的のエントリーを検索する際の負荷を軽減できるのは、第２の発明における第一の形式の場合と同様である。

【０２１１】

このような形式を採用すると、代替キー・ロケーション・テーブルＡＡＬＣで、該当代替キー・ブロック１７及び、当該代替キー・ブロック１７に連結されている代替キー・オーバーフロー・ブロック１５，１６の代替キー値の最小値及び／または最大値を保持することができることになる。これにより、代替キー・ブロックＡＡＣでは、自身と自身に連結されている代替キー・オーバーフロー・ブロック１５，１６に格納されている代替キー・エントリーの最小値と最大値を保持する必要がなくなり、代替キー・ブロック１７と代替キー・オーバーフロー・ブロック１５，１６の形式を同一とすることが可能となる。

【０２１２】

＜代替キー・ロケーション・テーブル使用時の効果＞

本第３の実施の形態に係るデータ及びデータベースの無停止自動再編成システ

ムでは、上述したような形式にすることで、次のような効果が期待できる。

まず第１に、再編成時の代替キー・ブロックと代替キー・オーバーフロー・ブロックの形式変更を変更する必要がなくなり、再編成時の負荷の削減が可能となる。

第２に、第一の形式を採用した場合には、再編成を行う際に、代替キー・ブロック、及び、代替キー・オーバーフロー・ブロックを現用代替キー・テーブルから、現用代替キー・テーブルへ書き移す必要があったが、この形式の場合には、後述するように、ロケーション・テーブルとブロックの再編成と同様なロジックとなるため、代替キー・ロケーション・テーブルの書き移しは必要であるが、代替キー・ブロック及び、代替キー・オーバーフロー・ブロックの書き移しが最小限で済むことになり、その面からも再編成時の負荷の削減が可能となる。

【０２１３】

第３に、第２の理由で述べたように、再編成時には現用代替キー・ロケーション・テーブルから、新規の代替キー・ロケーション・テーブルへの書き移しを行うため、現用と新規の代替キー・ロケーション・テーブルを保持する必要があるが、第一の形式を採用した場合には、現用と新規の両方の代替キー・テーブル用の領域が必要であることに比較すると、小さな領域で済む。

【０２１４】

第４に、第一の形式の場合には、代替キー・テーブルを連続領域に取得する必要があったが、本形式を採用して代替キー・ロケーション・テーブルを使用する場合には、代替キー・ロケーション・テーブルを連続領域に取得すれば、代替キー・ブロックは不連続領域に散在していても不都合は無いため、領域取得が柔軟に行える。一般的な条件下では、代替キー・ブロックに対して、代替キー・ロケーション・テーブルは数十分の１以下の大きさとなることが想定される。

【０２１５】

一方、代替キー・ロケーション・テーブルの領域の分だけ、常に余分な領域が必要となるのは、この方式のデメリットである。

[代替キー・テーブルが第二の形式である場合の再編成]

次に、本第３の発明の実施の形態に係るデータ及びデータベースの無停止自動

再編成システムによる代替キー・テーブルの第二の形式を採用した場合の再編成の方法に関して説明を、図１４を参照しながら行う。

【０２１６】

ここに、図１４は、本第３の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムの再編成の方法を説明するための図である。

この図１４において、本第３の発明の実施の形態による代替キー・テーブルの第二の形式とは、代替キー・テーブル１１Ａの代替キー・ブロック１７に対して、代替キー・ロケーション・テーブルＡＡＬＣを設けて、代替キー・ロケーション・テーブルＡＡＬＣの各エントリーが、代替キー・ブロック１７を管理するという形態である。代替キー・オーバーフロー・ブロック１５，１６が代替キー・ブロック・テーブルとブロックの関係と、殆ど同様である。

【０２１７】

＜２つの代替キー・ロケーション・テーブルと２つの再編成ポインター。代替キー・ロケーション・テーブルのエントリーの書き移し＞

よって、この場合の再編成に関しては、ロケーション・テーブルとブロックの再編成で述べた説明と、ほぼ同等のロジックで実行可能である。

【０２１８】

［代替キー・テーブルが第二の形式である場合の再編成：再編成時の必要領域］＜２つのロケーション・テーブル＞

この自動再編成システムによる自動再編成を実施する場合には、一時点で、再編成対象の代替キー・ロケーション・テーブルを２つ持つ必要があるため、領域の余裕が必要であるが、第２の発明（第一の形式）を採用した場合の、代替キー・テーブルを２つ持つ形式と比較すると、本第３の発明のほうが、二重化するのに必要な領域は、はるかに小さいもので済む。

【０２１９】

［代替キー・テーブルが第二の形式である場合の再編成：オーバーフローの解消］

図１２における代替キー・テーブル１１Ａが再編成対象となった場合に関して説明する。既に説明したように、代替キー・テーブル１１Ｂ、１１Ｃに関する説

明は省略するが、代替キー・テーブル１１Ａの場合と全く同様である。

【０２２０】

図１２、図１３及び図１４において、代替キー・ロケーション・テーブルＡＡＬＣ，ＡＡＬＮを用いた再編成とは、次に述べるようなことである。代替キー・ロケーション・テーブルＡＡＬＣが管理しているのは、代替キー・ブロック１７のみである。代替キー・オーバーフロー・ブロック１５，１６は代替キー・ブロック１７により管理されている。言い換えれば、代替キー・オーバーフロー・ブロック１５のアドレスを代替キー・ブロック１７が保持している、ということである。このため、代替キー・ロケーション・テーブルＡＡＬＣを利用して代替キーで検索を行う場合に、対象となる代替キー・ブロック１７を代替キー・ロケーション・テーブルＡＡＬＣに対してバイナリー・サーチを行うことにより探すが、更に代替キー・ブロック１７内では、対象となる代替キー・エントリーを探す必要がある。代替キー・ブロック１７に対して代替キー・オーバーフロー・ブロック１５，１６が多数連結されていると、対象となる代替キー・エントリーを探すための時間が、代替キー・ブロック１７しか存在しない場合に比較して、代替キー・オーバーフロー・ブロック１５，１６の数に関連して、余分に掛かることになる。これを避けるために、代替キー・オーバーフロー・ブロック１５，１６を無くして、総ての代替キー・ブロックをロケーション・テーブルＡＡＬＣで管理するようにすれば、対象代替キー・エントリーを探す時間が短縮できることになる。

【０２２１】

また、代替キー・エントリーは、代替キー・ブロック１７及び代替キー・オーバーフロー・ブロック１５，１６の中で、代替キーの順番に並べることになっており、代替キー・エントリーの挿入時に、代替キー・オーバーフロー・ブロックの数が多いと、代替キー・エントリーの移動が増加してしまい、挿入時の効率が低下するが、それを防止する目的もある。

【０２２２】

＜現用と新規の２つのロケーション・テーブルを使用して再編成を行う＞

代替キー・オーバーフロー・ブロック１５，１６が幾つ発生しているかは、数

えてあるので、代替キー・ロケーション・テーブルＡＡＬＣのエントリー数と、オーバーフロー・ブロック１５，１６の数を足したものが、新しいロケーション・テーブルＡＡＬＮのエントリー数となる。再編成時点で使用しているロケーション・テーブルを、ＡＡＬＮと対比するため、ＡＡＬＣと呼ぶことにする。エントリーは、再編成中に増加する可能性があり、再編成後もレコードの追加に伴い、代替キー・エントリーが増加するので、必要数より多めに確保することが好ましい。

【０２２３】

しかしながら、後述するように、他方で、フラグメンテーションを解消することにより、必要代替キー・ブロック数が少なくなる場合もあるし、適正な初期格納率の確保によっても、必要代替キー・ブロック数は変動するので、格納されているエントリーの容量と適正な初期格納率を基に計算するのが、最も好ましい方法である。

【０２２４】

新規代替キー・ロケーション・テーブルＡＡＬＮの容量を満たす連続領域を、プライマリー・システムに確保する。

領域の確保ができたら、現用代替キー・ロケーション・テーブルＡＡＬＣ０から新規代替キー・ロケーション・テーブルＡＡＬＮに対して、そのエントリーを順次書き移していくのであるが、以下のような手順で行う。

【０２２５】

＜再編成ポインター＞

まず、再編成ポインターと言うものを作成する。これは、代替キー・ロケーション・テーブルの、どのエントリーまで書き移しが終了したかを指し示すもので、現用代替キー・ロケーション・テーブルＡＡＬＣ用の現用再編成ポインターＲＰＡＡＬＣと、新規代替キー・ロケーション・テーブルＡＡＬＮ用の新規再編成ポインターＲＰＡＡＬＮとの２つを用意する。現用再編成ポインターＲＰＡＡＬＣの初期値は、現用代替キー・ロケーション・テーブルＡＡＬＣの先頭番地とし、新規再編成ポインターＲＰＡＡＬＮの初期値は新規代替キー・ロケーション・テーブルＡＡＬＮの先頭番地とする。

【０２２６】

次に、現用再編成ポインターＲＰＡＡＬＣ０の1番目のエントリーと、そのエントリーで管理している代替キー・ブロック１７及び代替キー・オーバーフロー・ブロックを排他する。この場合は代替キー・オーバーフロー・ブロックが存在しないため代替キー・ブロック１７（代替キー・ブロック番号「０」）のみとなる。

次に、1番目のエントリー（ブロック番号「０」）を現用代替キー・ロケーション・テーブルＡＡＬＣから新規代替キー・ロケーション・テーブルＡＡＬＮに書き移す（図１４のＳ５１）。その際に、1番目のエントリーで管理しているブロックに、オーバーフロー・ブロックが連結されているか否かを確認する。連結されていない場合は、現用再編成ポインターＲＰＡＡＬＣと新規再編成ポインターＲＰＡＡＬＮのアドレスを２番目のエントリーの先頭を指し示すように変更する。

【０２２７】

図１４では代替キー・オーバーフロー・ブロックが連結されていないので、現用再編成ポインターＲＰＡＡＬＣを現用代替キー・ロケーション・テーブルＡＡＬＣの２番目のエントリーを指し示すように変更する。同様に新規再編成ポインターＲＰＡＡＬＮを新規代替キー・ロケーション・テーブルＡＡＬＮの２番目のエントリーを指し示すように変更する。次に排他を行っていた場合には、ここで、１番目のエントリーの排他を解除する。

【０２２８】

次に、現用代替キー・ロケーション・テーブルＡＡＬＣの２番目のエントリーを排他する。２番目のエントリーには、２つの代替キー・オーバーフロー・ブロック１５，１６が連結されているので、代替キー・ブロック１７と、２つの代替キー・オーバーフロー・ブロックの排他を行う。

次に、２番目のエントリー（ブロック番号：１）の処理を行う。ＡＡＬＣ０の２番目のエントリーで管理されている代替キー・ブロックには、代替キー・オーバーフロー・ブロックが２つ連結されている。代替キー・オーバーフロー・ブロックが連結されている場合には、次の様にする。２番目の現用代替キー・ロケーシ

ョン・テーブルＡＡＬＣのエントリーを新規代替キー・ロケーション・テーブルＡＡＬＮの２番目のエントリーに書き移すのであるが、ブロックに格納されているエントリーの代替キー値の最小値と最大値に関する変更を行う。

【０２２９】

これは、現用代替キー・ロケーション・テーブルＡＡＬＣのエントリーに、代替キー・ブロック１７の代替キー値の最大値、最小値を持っている場合、単に書き移すだけでは、代替キー・ロケーション・テーブルＡＡＬＣのエントリーの代替キー値の最小値と最大値が、代替キー・ブロック１７（の代替キー・ブロック番号「１」）に格納されているエントリーの、代替キー値の最小値と最大値と合致しなくなってしまうため、これを回避するためである。

仮に、新規代替キー・ロケーション・テーブルＡＡＬＮの２番目のエントリーの代替キー値の最小値が０００００、最大値が０２９９だとする。そして、代替キー・ブロック１７（の代替キー・ブロック番号「１」）に格納されているエントリーの代替キー値の最小値が０００００、最大値が００９９、１番目の代替キー・オーバーフロー・ブロック１５に格納されているエントリーの代替キー値の最小値が０１００、最大値が０１９９、２番目の代替キー・オーバーフロー・ブロック１６に格納されているエントリーの代替キー値の最小値が０２００、最大値が０２９９とする。

【０２３０】

新規代替キー・ロケーション・テーブルＡＡＬＮの２番目のエントリーの代替キー値は、最小が００００、最大が００９９となる（図１４のＳ５２）。新規代替キー・ロケーション・テーブルＡＡＬＮのエントリーで保持される代替キー・ブロックのアドレスは、代替キー・ブロック１７の（代替キー・ブロック番号「１」）のアドレスと同じ値が入ることになる。

【０２３１】

次に、新規代替キー・ロケーション・テーブルＡＡＬＮの３番目のエントリーには、１番目の代替キー・オーバーフロー・ブロック１５のアドレスと、代替キー値の最小値には０１００、最大値には０１９９を入れる（図１４のＳ５３）。新規代替キー・ロケーション・テーブルＡＡＬＮの４番目のエントリーには、２

番目の代替キー・オーバーフロー・ブロック１６のアドレスを入れ、代替キー値の最小値は０２００、最大値は０２９９とする（図１４のＳ５４）。

【０２３２】

次に、代替キー・ブロック１７（代替キー・ブロック番号「１」）の中のオーバーフロー・ブロック・アドレスをリセットして、代替キー・オーバーフロー・ブロック１５が連結されていない状態にする（図１４のＳ５５）。次に１番目の代替キー・オーバーフロー・ブロック１５中の代替キー・オーバーフロー・ブロック・アドレスをリセットして、２番目の代替キー・オーバーフロー・ブロック１６が連結されていない状態にする（図１４のＳ５６）。

新規代替キー・ロケーション・テーブルＡＡＬＮの一つエントリーに関して、再編成が完了したら、そのエントリーによって管理されるブロックの排他を解除する。

【０２３３】

このようにすることで、代替キー・オーバーフロー・ブロックを、別の場所に書き換えることなく、再編成を行うことができる。

以下、同様に、現用代替キー・ロケーション・テーブルＡＡＬＣの３番目（代替キー・ブロック番号「２」）の再編成をし（図１４のＳ５７，Ｓ５８，Ｓ５９）、４番目（代替キー・ブロック番号「３」）のエントリーの再編成をし（図１４のＳ６０）、というように順次再編成を行っていく。

【０２３４】

図１４は現用代替キー・ロケーション・テーブルＡＡＬＣの４番目のエントリーまで、再編成が完了した状態を示している。

次に、新規再編成ポインターＲＰＡＡＬＮの値は、新規代替キー・ロケーション・テーブルＡＡＬＮの４番目のエントリーの先頭を指すように書き換える（図１４のＳ６１）。

【０２３５】

[代替キー・テーブルが第二の形式である場合の再編成：オーバーフローの解消の例外]

代替キー・オーバーフロー・ブロックの解消に関して述べたが、解消ができな

い場合に関して説明する。この解消ができない場合は、同一代替キーを保有するエントリーが多くて、一つのブロックに格納できない場合である。同一の代替キーを持つエントリーは、複数の代替キー・ブロックに格納されることは無く、必ず一つの代替キー・ブロックと１つ以上の代替キー・オーバーフロー・ブロック、または、複数の代替キー・オーバーフロー・ブロックに格納される。複数の代替キー・オーバーフロー・ブロックに格納されるとは、代替キー・ブロックとその直後の代替キー・オーバーフロー・ブロックには既にエントリーが格納されており、代替キー・オーバーフロー・ブロックの途中から同一代替キーのエントリーの格納が開始されるような場合である。

【０２３６】

同一代替キーのエントリーが格納されている場合は、代替キー・ブロック及び代替キー・オーバーフロー・ブロックにその情報を保持するのが分かりやすい。同一代替キーのエントリーの先頭部分か、中間部分か、最後尾部分か、といった情報を保持するのである。それ以外は、通常のエントリーを格納する場合と同様である。

この場合の、代替キー・オーバーフロー・ブロックの解消は、通常のエントリーの部分に関しては、解消を行えるが、同一代替キーのエントリーが格納される代替キー・ブロック及び代替キー・オーバーフロー・ブロックに関しては、必ず代替キー・オーバーフロー・ブロックを伴った形式となるため、代替キー・オーバーフロー・ブロックの解消はできない。このような場合には、再編成の情報として、同一代替キーのエントリーに関するものを出力することが好ましい。

【０２３７】

[代替キー・テーブルが第二の形式である場合の再編成：フラグメンテーションの解消]

上記では、代替キー・オーバーフロー・ブロックを解消する方法に関して述べたが、オーバーフローと同様にフラグメンテーションも、効率上では大きな問題であった。

これを解消するためには、代替キー・オーバーフロー・ブロックを解消する再編成と似た動作で実現できる。図１５を用いて説明する。ここに、図１５は、本

第３の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいて、フラグメンテーションを解消する方法を説明するための図である。

この図１５においても、代替キー・ロケーション・テーブルを使用するのであるが、現用代替キー・ロケーション・テーブルＡＡＬＣの他に、新規代替キー・ロケーション・テーブルＡＡＬＮを使用する。

図１５では、代替キー・ブロック１７において、代替キー・ブロック番号「０」、「１」、「１－２」、「１－３」、「２」、「２－２」には、適度にエントリーが格納されており、代替キー・オーバーフロー・ブロックの解消を前述の方法で完了したとする。

【０２３８】

以下の説明で使用するブロック番号は、現用代替キー・ロケーション・テーブルＡＡＬＣに基づく番号を使用する。

新規代替キー・ロケーション・テーブルＡＡＬＮに基づく番号を使用する場合には、その旨を記述する。

【０２３９】

代替キー・ブロック１７において、代替キー・ブロック番号「３」には、格納容量の３０％のエントリーが格納されている。代替キー・ブロック番号「４」には格納容量の４０％のエントリーが格納されている。代替キー・ブロック番号「５」は７０％、代替キー・オーバーフロー・ブロック１５の代替キー・ブロック番号「５－２」には６０％、代替キー・ブロック１７において代替キー・ブロック番号「６」には７０％のエントリーが格納されている。また、再編成後の各ブロックの格納率は９０％とする。これは、再編成後に挿入レコードが発生した場合に、その都度代替キー・オーバーフロー・ブロックを発生させないために設けることができる。

【０２４０】

本第３の発明の実施の形態に係る再編成システムは、新規代替キー・ロケーション・テーブルＡＡＬＮの新規再編成ポインターＲＰＡＡＬＮが指している代替キー・ブロック１７の代替キー・ブロック番号「３」の再編成を行おうとする。

しかしながら、格納率（代替キー・ブロックの容量に対する、その代替キー・ブロックに格納されているエントリーの容量の割合）が３０％であるため、適正な初期格納率を満たしていない。このため、代替キー・ブロック１７の次の代替キー・ブロック番号「４」を見る。これも格納率が４０％であるため、２つの代替キー・ブロックを足しても適正な初期格納率（９０％）を下回ることになる。更に代替キー・ブロック１７の代替キー・ブロック番号「５」を見ると格納率は７０％であるため、適正な初期格納率９０％を上回ることになる。代替キー・ブロック１７の代替キー・ブロック番号「３」に格納されているエントリーはそのまま残し、代替キー・ブロック１７の代替キー・ブロック番号「４」に入っていたエントリーを代替キー・ブロック１７の代替キー・ブロック番号「３」にする。これにより、代替キー・ブロック１７の代替キー・ブロック番号「３」は格納率が７０％となる。適正な初期格納率にするため、代替キー・ブロック１７の代替キー・ブロック番号「５」に格納されているエントリーの先頭２０％を代替キー・ブロック１７の代替キー・ブロック番号　「３」に移動すると同時に、残りのエントリー５０％を前方（図示左側）に詰める。代替キー・ブロック１７の代替キー・ブロック番号「３」が完成したので、新規再編成ポインターＲＰＡＡＬＮの代替キー・ブロック１７の代替キー・ブロック番号「６」のアドレスを代替キー・ブロック１７の代替キー・ブロック番号「３」に書き換える。新規再編成ポインターＲＰＡＡＬＮを新規代替キー・ロケーション・テーブルＡＡＬＮの代替キー・ブロック１７の代替キー・ブロック番号「７」の先頭に移動する。この時点で代替キー・ブロック１７の代替キー・ブロック番号「４」は使用されない代替キー・ブロックとなる。

【０２４１】

次に、代替キー・ブロック１７の代替キー・ブロック番号「５」を操作するのであるが、再編成により代替キー・ブロック１７の代替キー・ブロック番号「５」は格納率が５０％となっている。代替キー・オーバーフロー・ブロック１５の代替キー・ブロック番号「５－２」は格納率が６０％であるので、代替キー・ブロック番号「５－２」のエントリーの内、先頭から３０％のエントリーを代替キー・ブロック１７の代替キー・ブロック番号「５」に移動し、同時にブロック番

号「５－２」に残っているエントリーを前方（図示左側）に詰める。更に代替キー・ブロック１７の代替キー・ブロック番号「５」が代替キー・ブロック１７の代替キー・ブロック番号「５－２」を代替キー・オーバーフロー・ブロックとして連結しているのを切断する。これは、代替キー・ブロック１７の代替キー・ブロック番号「５」の代替キー・オーバーフロー・ブロック・アドレスを特定の値（例えばゼロ）にセットする。代替キー・ブロック１７の代替キー・ブロック番号「５」は完成したので、新規代替キー・ロケーション・テーブルＡＡＬＮの代替キー・ブロック１７の代替キー・ブロック番号「７」のアドレスを代替キー・ブロック１７の代替キー・ブロック番号「５」に書き換える。そして、新規再編成ポインターＲＰＡＡＬＮを新規代替キー・ロケーション・テーブルＡＡＬＮ０の代替キー・ブロック１７の代替キー・ブロック番号「８」の先頭に移動する。

【０２４２】

次に、代替キー・オーバーフロー・ブロック１５の代替キー・ブロック番号「５－２」の格納率が３０％であるので、次の代替キー・ブロック１７の代替キー・ブロック番号「６」を見ると格納率が６０％であるので、代替キー・ブロック１７の代替キー・ブロック番号「６」のエントリー総てを代替キー・ブロック１７の代替キー・ブロック番号　「５－２」に移動する。代替キー・ブロック１７の代替キー・ブロック番「５－２」が完成したので、新規代替キー・ロケーション・テーブルＡＡＬＮの代替キー・ブロック１７の代替キー・ブロック番号「８」のアドレスを代替キー・ブロック１７の代替キー・ブロック番号「５－２」に書き換える。新規再編成ポインターＲＰＡＡＬＮを新規代替キー・ロケーション・テーブルＡＡＬＮの代替キー・ブロック番号「９」の先頭に移動する。この時点で代替キー・ブロック１７の代替キー・ブロック番号「６」は使用されない代替キー・ブロックとなる。

【０２４３】

[代替キー・テーブルが第二の形式である場合の再編成：適正な初期格納率の確保]

また、フラグメンテーションとは逆に、適正な初期格納率を確保するために、代替キー・ブロックの追加が必要な場合も発生する。これは、例えば、すべての

代替キー・ブロックの格納率が１００％であった場合に、適正な初期格納率を９０％とすると、一時点で９個の代替キー・ブロックを再編成の対象にし、１つの代替キー・ブロックを追加して１０個の代替キー・ブロックに対して、エントリーを９０％ずつ格納する、というような形態である。

適正な初期格納率の確保を厳密に行おうとする、代替キー・ブロックとエントリーの大きさの関係で不可能な場合や、一時点でかなり多数の代替キー・ブロックを再編成の対象にしなければならなくなる可能性があるため、排他範囲が広がり、システムの稼動に悪影響を与える可能性がある。

【０２４４】

このような状態が発生することを防止する上で、適正な初期格納率を、例えば、８５％から９０％など、複数の値を用いることが、運用上は好ましい。この場合、適正な初期格納率が、８５％から９０％の範囲に入っていればよい、とするものである。

この適正な初期格納率の確保を行う場合にも、エントリーが元の代替キー・ブロックから別の代替キー・ブロックに書き換えられることが発生する。

【０２４５】

[代替キー・テーブルが第二の形式である場合の再編成：実際の再編成とデッドロックの防止]

実際に再編成を行う場合には、代替キー・オーバーフロー・ブロックの解消と、フラグメンテーションの解消、及び、適正な初期格納率の確保を同時に実行することになるので、上記の３つの方式を組み合わせて使用する。

原理的には以上のような方法で実現が可能であるが、代替キー・ブロックを順次読んでいき、適正な初期格納率の代替キー・ブロックを作成していくと、当初から排他の範囲が確定せず、順次に排他していくために、デッドロックが発生する可能性が増加する。デッドロックの発生を防ぐためには、次のような方法が有効である。

第一の方法は、まず、排他を掛けずに次々と代替キー・ブロックを読みだし、レコードの格納率を調べて、複数のブロックを組み合わせて、適正な大きさとなるように計算をした上で、排他範囲を決定する方法である。

【０２４６】

＜ロケーション・テーブルのエントリーへの項目追加＞

第二の方法は、代替キー・ロケーション・テーブルのエントリーに、ブロック番号とブロックのアドレスの他、ブロックに格納されているレコードの主キー値の最小と最大の両方、または何れか一方に加えて、ブロック中のレコードの格納率若しくは、レコードが占めている領域のバイト数を、エントリーに追加する。このようにすることで、第一の方法で述べた、ブロックを読むという作業を必要とせず、代替キー・ロケーション・テーブルを読むだけで作業が行えることになる。しかしながら、この場合にはエントリーの挿入や削除に伴って、代替キー・ロケーション・テーブルの書き換えが発生することになるため、エントリーの発生状況に応じて、第一の方法と第二の方法のいずれかを選択することが好ましい。

【０２４７】

このようにして、代替キー・ブロック及び代替キー・オーバーフロー・ブロック内の格納率を調べた後で、その時点での再編成範囲を確定し、その範囲の代替キー・ロケーション・テーブルのエントリーと、そのエントリーが指している代替キー・ブロック及び、代替キー・ブロックに連結されている代替キー・オーバーフロー・ブロックの排他を行う。

【０２４８】

その後、再編成対象となる代替キー・ブロックと代替キー・オーバーフロー・ブロックの数と格納可能容量を調べ、格納対象エントリーの実際の容量を調べた上で、必要になる代替キー・ブロックが現用代替キー・ブロックと代替キー・オーバーフロー・ブロックの和と同数か、増加するか、減少するか、を調べる。この後、代替キー・オーバーフロー・ブロックの解消や、フラグメンテーションの解消、適正な初期格納率の確保で説明したロジックを使用して、代替キー・ブロックの再編成と、新規代替キー・ロケーション・テーブルのエントリーの作成を行う。

【０２４９】

この場合、再編成ポインターは、再編成の対象となったブロックの分だけ、一

度に移動することになる。

このように、一時点で、１個から数十個の代替キー・ロケーション・テーブル・エントリーと代替キー・ブロックを対象として再編成を実施するのであるが、この再編成は、見かけ上、通常のデータ処理のトランザクションとして扱うことにより、通常のデータ処理との矛盾を発生させない。
また、この再編成を次々と、代替キー・ロケーション・テーブルと代替キー・ブロックに対して実施することにより、それら全体の再編成を完了する。

【０２５０】

[代替キー・テーブルが第二の形式である場合の再編成：再編成の完了]

代替キー・ロケーション・テーブルの再編成が完了したことを認識する方法に関して説明する。現用代替キー・ロケーション・テーブルＡＡＬＣには、その代替キー・ロケーション・テーブルＡＡＬＣを使用している最終位置を示すための、代替キー・ラスト・ポインターが設けてある。代替キー・ラスト・ポインターを設ける目的は、次のようなものである。代替キー・ロケーション・テーブルＡＡＬＣは連続領域に確保することになっている。代替キー・ロケーション・テーブルのエントリーが不足した場合には、最初の代替キー・ロケーション・テーブルとは連続しない領域に追加の代替キー・ロケーション・テーブルを設けて、それらを連続した領域であるかのようにバイナリー・サーチできるよう、アドレス変換して検索することが可能であるが、不連続領域が多くなると、アドレス変換の負荷が増大するため、好ましくない。このため、代替キー・ロケーション・テーブル用の領域は、予め十分に大きな領域を確保しておき、使用しているエントリーと、未使用のエントリーを区分するために、使用しているエントリーの次のエントリーのアドレスを指す、ということになっている。

【０２５１】

このように、代替キー・ラスト・ポインターを設けることにより、現用代替キー・ロケーション・テーブルＡＡＬＣの先頭アドレスと代替キー・ラスト・ポインターの間でバイナリー・サーチを実行することにより、現用代替キー・ロケーション・テーブルＡＡＬＣに未使用エントリーが存在していても、主キーによるレコード検索が可能となる。

【０２５２】

＜再編成の完了の検出方法＞

この代替キー・ラスト・ポインターを指標として用いる。再編成を実行して行き、現用再編成ポインターＲＰＡＡＬＣが指すアドレスが、代替キー・ラスト・ポインターが指すアドレスと一致した場合に、代替キー・ロケーション・テーブルＡＡＬＣと代替キー・ブロック１７の再編成が完了したことになる。尚、最終エントリーが指している代替キー・ブロック１７に、代替キー・オーバーフロー・ブロックが連結されている場合でも、それらの、代替キー・オーバーフロー・ブロックの再編成が完了しない限り、現用代替キー・ロケーション・テーブルＡＡＬＣの移動は行われないので、問題ない。

【０２５３】

[代替キー・テーブルが第二の形式である場合の再編成：未使用ブロックの再利用]

このようにして、フラグメンテーションに対する再編成の方法に関して述べたが、図１５の場合には、現用代替キー・ロケーション・テーブルＡＡＬＣの代替キー・ブロック１７の代替キー・ブロック番号「４」と代替キー・ブロック１７の代替キー・ブロック番号「６」が使用されなくなってしまった。このままでは、代替キー・ブロック内のフラグメンテーションは解消されるが、全体的なフラグメンテーションは解消されない、ということになりかねない。このようなことを防止するために、つぎのような方式を採用する。

【０２５４】

＜未使用ブロック・アロケーション・テーブル。開始位置ポインター、終了位置ポインター＞

図１６は、本第３の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいて、全体的なフラグメンテーションを解消する方法について説明するための図である。

この図１６における実施の形態では、未使用代替キー・ブロック・アロケーション・テーブルＵＢＡＴというものを使用する。この未使用代替キー・ブロック・アロケーション・テーブルＵＡＢＡＴは、図１６に示すような形式のテーブル

であって、未使用代替キー・ブロックのアドレスを格納しておくためのものである。未使用代替キー・ブロック・アロケーション・テーブルＵＡＢＡＴの開始位置を示すポインター（ＮＡＡＡＢＰＳ）と終了位置を示すポインター（ＮＡＡＡＢＰＥ）の２つを使用する。図１６では、未使用代替キー・ブロックが全く無い状態から、７つの未使用代替キー・ブロックが発生した後の状態を示している。

【０２５５】

初期状態では、開始位置を示すポインターＮＡＡＡＢＰＳと、終了位置を示すポインターＮＡＡＡＢＰＥは、両方とも未使用代替キー・ブロック・アロケーション・テーブルの先頭を指している。未使用代替キー・ブロックが発生する（図１６では「未使用」と表示されている）と、未使用代替キー・ブロック・アロケーション・テーブルＵＢＡＴの終了位置を示すポインターＮＡＡＡＢＰＥの指しているエントリーに、未使用代替キー・ブロックのアドレスを登録し、終了位置を示すポインターＮＡＡＡＢＰＥを未使用代替キー・ブロック・アロケーション・テーブルＵＡＢＡＴの次のエントリーを指すように書き換える。順次、７つの未使用代替キー・ブロックが発生した後の状態が図１６に示したものである。ここでは、終了位置を示すポインターＮＡＡＡＢＰＥは、図１６に示すように、未使用代替キー・ブロック・アロケーション・テーブルの８番目のエントリーを指している。

【０２５６】

図１７は、本第３の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいて、ブロックの再利用を説明するための図である。

この図１７を用いて、再利用の方法について説明する。新たに代替キー・ブロックまたは代替キー・オーバーフロー・ブロックを取得する必要が発生した場合、本第３の発明の実施の形態では、未使用領域から代替キー・ブロックを取得するのではなく、未使用代替キー・ブロック・アロケーション・テーブルＵＡＢＡＴを参照し、未使用代替キー・ブロックが存在する場合には、未使用代替キー・ブロック・アロケーション・テーブルＵＡＢＡＴに登録してある代替キー・ブロックを優先して使用する。使用する方法は、開始位置を示すポインターＮＡＡＡＢＰＳが指しているエントリーの代替キー・ブロックを使用する。エントリーに

は未使用代替キー・ブロックのアドレスが入っているので、このアドレスを、代替キー・ブロック追加の場合は代替キー・ロケーション・テーブルに書込み、代替キー・オーバーフロー・ブロックの場合は、その代替キー・ブロックを管理する代替キー・ブロックまたは代替キー・オーバーフロー・ブロックのポインターに、そのアドレスを書き込む。その後、開始位置を示すポインターＮＡＡＡＢＰＳの内容を未使用代替キー・ブロック・アロケーション・テーブルの次のエントリーを指すように書き換える。

【０２５７】

上述したように書き換えた結果、２つの未使用ブロックが使用された直後の状態を示したものが図１７である。

この未使用代替キー・ブロック・アロケーション・テーブルＵＡＢＡＴは、循環使用が可能である。未使用代替キー・ブロックが発生すると、終了位置を示すポインターＮＡＡＡＢＰＥの位置は、未使用代替キー・ブロック・アロケーション・テーブルＵＡＢＡＴの後方に動いていくが、一方で、未使用代替キー・ブロックが使用されると、開始位置を示すポインターＮＡＡＡＢＰＳの位置も後方にずれていくので、終了位置を示すポインターＮＡＡＡＢＰＥが開始位置を示すポインターＮＡＡＡＢＰＳを追い越さない限り、一つのテーブルＵＡＢＡＴを循環して使用する、つまり、終了位置を示すポインターＮＡＡＡＢＰＥが未使用代替キー・ブロック・アロケーション・テーブルＵＡＢＡＴの最後まで行った場合に、また、未使用代替キー・ブロック・アロケーション・テーブルＵＡＢＡＴの先頭に戻って当該テーブルＵＡＢＡＴを再使用することが可能である。

【０２５８】

［代替キー・テーブルが第二の形式である場合の再編成：再編成中のデータベースのアクセス］

次に、再編成中の、データの検索・読み書きに関して、再編成を行っているときにも、データの検索・読み書きが可能であることを、図１５を参照しながら述べる。

代替キー値による検索では、現用代替キー・ロケーション・テーブルＡＡＬＣを用いてバイナリー・サーチを行う。この際に、ターゲット・キー値（検索の対

象となるキー値）が再編成ポインターＲＰＡＡＬＣの上方にあるか、下方にあるかの判定を行う。これは、現用代替キー・ロケーション・テーブルＡＡＬＣが主キー値の順に並んでいることから、現用再編成ポインターＲＰＡＡＬＣの示しているエントリーのキー値とターゲット・キー値の比較を行えばよい。

【０２５９】

ターゲット・キー値が、現用再編成ポインターＲＰＡＡＬＣが指しているエントリーの代替キー値の最小値より小さい場合は、現用再編成ポインターＲＰＡＡＬＣより上位（アドレスが小さい）に、目的とするエントリーが存在することになる。この場合は、新規代替キー・ロケーション・テーブルＡＡＬＮを使用して、新規代替キー・ロケーション・テーブルＡＡＬＮの先頭アドレスと新規再編成ポインターＲＰＡＡＬＮ間でバイナリー・サーチを行う。バイナリー・サーチの結果、目的のエントリーが指している代替キー・ブロックの中のエントリーを調べて、目的のエントリーが存在するか否かを検出する。

ターゲット・キー値が現用再編成ポインターＲＰＡＡＬＣの指しているエントリーの代替キー値の最小値以上である場合には、目的とするエントリーは、現用再編成ポインターＲＰＡＡＬＣより下位（アドレスが大きい、かつ、ＲＰＡＡＬＣ０が指しているエントリー）に存在することになる。この場合は、現用代替キー・ロケーション・テーブルＡＡＬＣを使用して、現用再編成ポインターＲＰＡＡＬＣと現用代替キー・ロケーション・テーブルＡＡＬＣのラスト・ポインター間のエントリーに対してバイナリーサーチを行う。

【０２６０】

このように、目的のエントリーの検索が確実に実行できるので、そのエントリーが指しているブロック内を検索することにより、目的とする主キー値を持つレコードの検索が可能となる。

上記の説明では、代替キー・テーブルを再編成中である場合に関して述べた。複数のテーブルを同時に再編成しないので、代替キー・テーブルの再編成中である場合には、ロケーション・テーブルは再編成中では無い、ということになる。これにより、ロケーション・テーブルとブロックの再編成で述べたような、現用と新規のロケーション・テーブルが２つ存在する、ということはないので、何れを

使用するかの工夫は不要である。

【０２６１】

また、目的とする代替キー値を持つエントリーが含まれる代替キー・ブロックが、正に再編成中で、排他されている場合には排他待ちとなり、その排他が解除されるまでは、当該代替キー・ブロックのアクセスはできないことになるが、これは、通常のアクセスでレコードの更新や挿入、削除が行われる場合と同様である。言い換えれば、排他中の代替キー・ブロックに対しての要求は、排他待ちになるが、その代替キー・ブロックに対する再編成が完了して、排他が解除されれば、処理が可能となる。

【０２６２】

上記では、レコード検索の場合に関して述べたが、これを応用することで、代替キーによる、レコードの更新、削除を行うことも可能である。レコードの挿入は、主キーで実行するので、ここでは対象外となる。また、レコードの削除は代替キーがノン・ユニークであるため、実行する場合には注意が必要である。しかしながら、この場合には、次に述べるような可能性がある。

【０２６３】

[代替キー・テーブルが第二の形式である場合の再編成：検索中に再編成が進行した場合への対処]

図１８は、本第３の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいて、代替キーの検索を行っている最中に再編成が進んでしまい、検索開始時の現用再編成ポインターと検索終了時の現用再編成ポインターの位置が異なっている場合の動作についての説明図である。

【０２６４】

上述したように再編成ポインターを使用し、ターゲット・キー値と比較することで、現用代替キー・ロケーション・テーブルＡＡＬＣ、新規代替キー・ロケーション・テーブルＡＡＬＮの何れかを使用するかを決定することにより、再編成中でもエントリーの呼び出しが可能であることを説明した。

しかしながら、図１８に示すように、代替キー・ロケーション・テーブルＡＡＬＣやＡＡＬＮを用いて、代替キーの検索を行っている最中に再編成が進んでし

まい、検索開始時の現用再編成ポインターＲＰＡＡＬＣと検索終了時の現用再編成ポインターＲＰＡＡＬＣの位置が異なっていて、その範囲の代替キー・ブロックが検索対象となった場合には、代替キー・オーバーフロー・ブロックの連結が、既に切れているために、実際には存在するエントリーが、存在しないことになってしまう可能性がある。図１８では、プライマリー・ブロック「５」に連結されていたオーバーフロー・ブロック１５が切り離されてしまっている。このままでは、不安定な動作を引き起こすことになり、使用できない。

【０２６５】

これは、次のような工夫を行うことにより、問題なく検索することが可能である。次の表にまとめたが、ターゲット・キー値と再編成ポインターによって、検索対象となるロケーション・テーブルが現用代替キー・ロケーション・テーブルＡＡＬＣであるか、新規代替キー・ロケーション・テーブルＡＡＬＮ０であるかの判定を行う。使用する代替キー・ロケーション・テーブルがＡＡＬＮ０である場合には、新規再編成ポインターＲＰＡＡＬＮが検索開始時よりも進んでも、問題は発生しない。問題は現用代替キー・ロケーション・テーブルＡＡＬＣを対象とした場合である。サーチの対象が現用代替キー・ロケーション・テーブルＡＡＬＣの場合で、現用再編成ポインターＲＰＡＡＬＣの移動がないときには、問題は発生しない。一方、サーチの対象が現用代替キー・ロケーション・テーブルＡＡＬＣの場合で、現用再編成ポインターＲＰＡＡＬＣの移動があるときには、工夫をしないと問題が発生することになる。

【０２６６】

＜再編成が進行した場合のアクセス方法＞

検索を開始する前に、再編成中であれば、現用再編成ポインターＲＰＡＡＬＣの値（代替キー・ロケーション・テーブルで、次に再編成を行うエントリーのアドレス）と新規再編成ポインターＲＰＡＡＬＮの値を保存しておく。これらを、Ｓ－ＲＰＡＡＬＣと、Ｓ－ＲＰＡＡＬＮとする。そして、現用代替キー・ロケーション・テーブルＡＡＬＣの検索を完了した時点で、その時点の現用再編成ポインターＲＰＡＡＬＣの値（これを、Ｅ－ＲＰＡＡＬＣ０とする）とＳ－ＲＰＡＡＬＣ０の値を比較する。この値が異なっていれば、検索中に再編成が進行したこ

とを意味する。この場合には、検索対象ブロックが何処であるかを判定する。もしも、Ｓ－ＲＰＡＡＬＣ０とＥ－ＲＰＡＡＬＣ０の間に対象代替キー・ブロックがある場合には、前述したように、エントリーが検索できない可能性がある。

【０２６７】

この場合には、新規代替キー・ロケーション・テーブルＡＡＬＮを使用して、Ｓ－ＲＰＡＡＬＮ０とＥ－ＲＰＡＡＬＮ０の間に対してバイナリー・サーチを行う。ここで、エントリーが発見できれば、エントリー有り、発見できなければエントリー無しとなる。このようにすることで、存在するエントリーが存在しないとなってしまう事象を避けることができる。

以上説明したように、再編中であってもエントリーの読み出しが可能であることがわかる。

【０２６８】

[代替キー・テーブルが第二の形式である場合の再編成：エントリーの挿入の場合]

上記説明では呼び出しに関して述べたが、レコード挿入などに伴うエントリー挿入の場合も、エントリーを挿入する代替キー・ブロックを決定するためには、バイナリー・サーチで代替キー・ブロックを見つけて、その後、当該代替キー・ブロックに対してエントリーを挿入する必要があり、呼び出しと同様の動作となる。

【０２６９】

レコードの代替キー値が変更される場合には、代替キー・エントリーの格納場所を変更する必要がある。何故ならば、代替キー・ブロックの中には、代替キー値の順序でエントリーが並ぶことになっており、その範囲を外れるエントリーが存在すると、代替キー・ロケーション・テーブルで検索が不能になってしまうからである。このため、代替キー値を変更する場合には、現エントリーを一旦削除し、その後、新しいエントリーを、その代替キー値に基づいて格納されるべき代替キー・ブロックを探して、そこに挿入する。この方法は、旧来のデータベースで一般的に使用されている方法である。

これ以外は、上記呼び出しと同様の方法で、代替キー・ブロックを探して書き

込むことが可能である。

【０２７０】

[再編成の中断と再開]

以上で説明したように、代替キー・テーブルの再編成中であっても、代替キー値による、レコードの検索（読み込み）や、更新、レコードの挿入、追加、削除が可能であることを説明した。つまり、再編成がどの時点、言い換えれば、代替キー・ロケーション・テーブルのどのエントリーまで進行していても、レコードのアクセスが可能であることはいうまでもない。

【０２７１】

＜再編成の中断と再開＞

これに基づけば、一時的に再編成を中断しても、レコードのアクセスは問題なく実行できることが分かる。勿論、中断は、あるまとまりの代替キー・ロケーション・テーブルと代替キー・ブロックに対する再編成が終了した後に行われることは当然である。

【０２７２】

本第３の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいて、再開は、中断した時点で、現用の再編成ポインターＲＰＡＡＬＣと新規の再編成ポインターＲＰＡＡＬＮが示している、現用の代替キー・ロケーション・テーブルＡＡＬＣと新規の代替キー・テーブルＡＡＬＮのエントリーから再編成を実施すればよい。

【０２７３】

この機能により、プライマリー・システムの負荷が増加した場合には、再編成を中断して、データ処理に能力を振り分け、データ処理の負荷が減少してきたら、再編成を再開することが可能となるので、再編成の実施に当たって、プライマリー・システムの負荷や運転状況に関して事前予測を行い、一定の時間を予め確保する必要がない。

代替キー・テーブルのエントリーに関しては、次の３種類が存在することを述べた。すなわち、ブロックの番号を保持する、ブロックのアドレスを保持する、ブロックの番号、アドレスの何れも保持しない、の３種類である。また、これら

については、以下の特徴があることも述べた。

ブロックの番号、アドレスの何れも保持しない場合には、再編成に要する負荷や時間が短縮できるが、代替キーを用いた検索では代替キー・エントリーを検索後にロケーション･テーブルの検索が必要になるため、検索の負荷が増加する。

一方で、ブロックの番号を保持する、ブロックのアドレスを保持する場合には、再編成時には番号やアドレスが変更になる場合に、情報の変更が必要になり、再編成の負荷が大きくなるが、一方で、代替キーを使用した検索は効率的となる。

【０２７４】

ここで、代替キー・エントリーの形式を、何れかに決定して代替キー・テーブルを作成して運用をしている場合、当初の予定と状況が変わってしまい、例えば、再編成が頻繁に必要になったとか、逆に、当初予定していた頻度に比較すると、再編成の頻度が著しく低下した場合、などが考えられる。

【０２７５】

このような場合、代替キー・テーブルの再編成時に、エントリーの形式を変更して、検索や追加・更新・削除等の効率は低下させても再編成の速度を向上させるようにする、または逆に、再編成の速度を低下させても検索や追加・更新・削除等の効率を高める、といった切り替えが出来ると好都合である。

これは、次のように代替キー･テーブルを再編成する場合に、代替キー・エントリーの形式を変更することにより可能となる。

ブロック番号やブロック・アドレスを保持しない形式の代替キー・エントリーに対して、ブロック番号やアドレスを付加する場合には、１代替キー・ブロック当たり、代替キー・ブロックの中に既に存在する代替キー・エントリーの数と、ブロック番号またはブロック・アドレスの分だけ情報量が増加することになる。

【０２７６】

一時点で、１個または複数個の代替キー・ブロックを対象にして、オーバーフローの解消、フラグメンテーションの解消、適正な初期格納率の確保を行う場合に、情報の増加分が書き込めるように代替キー・ブロックの確保を行う。対象となった代替キー・ブロックのエントリーに関しては、ブロック番号またはブロッ

ク・アドレスを追加し、エントリー情報の書き換えを行う。

この場合、対象となる代替キー・エントリー一つづつについて、そのエントリーの主キー値に基づいて、ロケーション･テーブルの検索を行い、見つかったブロックの番号、またはブロック・アドレスを新規の代替キー・テーブルのエントリーに付加する。

逆に、ブロック番号またはブロック・アドレスを削除する場合には、１代替キー・ブロック当たり、代替キー・ブロックの中に既に存在する代替キー・エントリーの数と、ブロック番号またはブロック・アドレスの分だけ情報量が減少することになる。

【０２７７】

一時点で、１個または複数個の代替キー・ブロックを対象にして、オーバーフローの解消、フラグメンテーションの解消、適正な初期格納率の確保を行う場合に、情報の減少分を考慮して代替キー・ブロックの確保を行う。対象となった代替キー・ブロックのエントリーに関しては、ブロック番号またはブロック・アドレス削除し、エントリー情報の書き換えを行う。付加する場合に比較すると、情報を削除するだけでよいので、時間的には短くて済む。

【０２７８】

[排他方法と排他方式]

ここで、排他方法と排他方式に関して説明を行う。排他方式とは、排他を実現するための方式であり、システムに対して負荷の低い方法を考案した。排他方法とは、主に排他の順序のことである。システム内において、排他の順序が異なるとデッド・ロックの原因になるため、排他順序を揃えることは重要である。本方式及び本システムにおける排他順序が同じであることを説明する。

【０２７９】

[排他方式]

排他方式は、ロケーション・テーブル、ブロック（オーバーフロー・ブロックを含む）、代替キー・ロケーション・テーブル、代替キー・ブロック（代替キー・オーバーフロー・ブロックを含む）のそれぞれに対して、直接的に排他している状態を書き込むことで実現する。

トランザクションが発生すると、キューと言う形でトランザクションが発生した順番に並べられ、その順序に従って処理が行われるのは、一般的な情報処理で用いられる形態である。更にトランザクション内でデータに対するアクセス要求をリクエスト・ブロックという形に表現するされることが、一般的である。トランザクションを実行する場合には、各種のデータに対するアクセスが発生する。同じ種類のデータに対するアクセスであっても、ロケーション・テーブルでの検索か、代替キー・テーブルでの検索かという違いがある。

【０２８０】

このリクエスト・ブロックには、処理要求発生元の識別、トランザクション番号、対象データ種別、処理要求識別（読み取り、更新、追加、挿入、削除、の種別）、読み取り対象キー、読み取り対象キー値、書込みレコード、などに関する情報が、そのまま、若しくは、それらの情報が入っているアドレスを指し示す、という形態で格納されている。

このリクエスト・ブロックに、２つのフィールドを追加する。１つは、排他テーブル・アドレスであり、もう一つは排他テーブル・ポインターである。排他テーブル・アドレスは、排他情報を格納するためのテーブルである。このテーブルのエントリーは、排他対象のアドレスを保持する他、次に述べる、アドレス識別フラッグを保持する。エントリーの大きさは、アドレスが保持できる大きさとする。テーブルの大きさ（エントリー数）は、例えば、１００個とし、それで不足する場合には、更に１００個を追加し、最初のテーブルの最後のアドレスが、次の排他テーブルのアドレスを保持するようにする。この場合、排他対象のアドレスか、次のテーブルのアドレスであるかを識別するために、アドレス識別フラッグを設ける。

【０２８１】

リクエスト・ブロックで、排他的なアクセス要求があった場合に、排他テーブルに排他対象のアドレスを入れる。ロケーション・テーブルの場合は当該エントリーのアドレスであり、ブロックの場合は当該ブロックのアドレス、代替キー・ロケーション・テーブルの場合は当該エントリーのアドレス、代替キー・ブロックまたは代替キー・オーバーフロー・ブロックの場合は当該ブロックのアドレス

となる。

【０２８２】

図１９は、本第１又は第３の発明に係るデータ及びデータベースの無停止自動再編成システムにおいて、ロケーション・テーブルを排他する場合の例を示す図である。この図１９において、符号１１０はリクエスト・キュー、符号１２０，１２１はリクエスト・ブロック、符号１３０－０は，１３０－１，１３１－０は排他テーブル、符号ＬＣはロケーション・テーブルである。

排他テーブル１３０－０のエントリー「０」は、ロケーション・テーブルＬＣのエントリー「０」を指している。これは、排他テーブル５０－０のエントリー「０」が、常にロケーション・テーブルＬＣのエントリー「０」を指すということではなく、例として、そのような対応関係であることを示している。ロケーション・テーブルＬＣには、排他テーブル１３０－０，１３０－１，１３１－１のエントリーのアドレスを保持できるように、フィールドを追加する。このロケーション・テーブルＬＣのエントリー「０」は、排他テーブル１３０－０のエントリー「０」を指している。

【０２８３】

ロケーション・テーブルＬＣのそのエントリーが、排他されているか否かは、そのエントリーの排他テーブル・エントリー・アドレス・フィールドにアドレスが入っているか否かを識別すればよい。アドレスが入っている場合には、そのエントリーは排他されているので、他の要求からは、アクセスできない。

排他テーブル１３０－０，１３０－１，１３１－０で、ロケーション・テーブルＬＣのエントリー・アドレスを保持しているのは、トランザクションが完了した場合に排他解除を確実に行うと共に、万一、システムに異常があり、全体の運用が停止してしまった場合に、排他解除を素早く実行するためのものである。

ロケーション・テーブルＬＣのエントリーに、排他されているか否かの識別を持っているが、排他テーブル（１３０－０，１３０－１，１３１－０）を持たない場合には、運用停止後に、総てのロケーション・テーブルＬＣのエントリーを見て、排他状態になっている場合には、排他を解除してやる必要がある。一方で、排他テーブル１３０－０，１３０－１又は１３１－０のエントリーにロケーシ

ョン・テーブルＬＣのアドレスが入っている場合には、それらの、排他テーブル１３０－０，１３０－１又は１３１－０を見て、該当するロケーション・テーブルＬＣのエントリーの排他を解除してやればよい。

排他テーブル・ポインターは、排他テーブル（１３０－０，１３０－１又は１３１－０）のどこまで使用しているかを指し示す目的で使用されている。図１９では、追加の排他テーブル１３０－１が確保されて、その途中まで使用されている状態を示している。追加の排他テーブル１３０－１の位置は、最初の排他テーブル１３０－０からポイントされる仕組としてある。

【０２８４】

［排他方法］

排他の順序は、デッドロックの発生と密接に関係する。アクセスの種類によって、排他の順序が異なっていると、そのことにより、デッドロックの発生する可能性が増大するので、アクセスの種類が異なっていても、同じ順序で排他する必要がある。検索系のアクセスの場合は、排他は必要ないため、ここでは除外する。更新系（追加、挿入、更新、削除）のアクセスの場合に関して述べる。

排他は、ロケーション・テーブル→ブロック→（代替キー・ロケーション・テーブル）→代替キー・ブロックという順序で行うこととする。代替キー・ロケーション・テーブルは、使用する方法と使用しない方法とがあるために括弧をつけてあるが、代替キー・ロケーション・テーブルを使用する場合には、代替キー・ロケーション・テーブルの排他を必ず実施する。排他の対象を、ここで例えば、ロケーション・テーブルとしてあるが、これはロケーション・テーブル全体ではなく、対象となるエントリーのみである。

【０２８５】

ロケーション・テーブル及び、代替キー・テーブル、代替キー・ロケーション・テーブルに対して、バイナリー・サーチで目的のエントリーを検索する際に、バイナリー・サーチの途中で、排他されているエントリーや代替キー・ブロックが２分割点に該当する場合が発生するが、この場合は、排他を無視してバイナリー・サーチを続行する。

【０２８６】

[主キーでのアクセスの場合の排他方法]

主キーでのアクセスは、まず、ロケーション・テーブルに対してバイナリー・サーチを行って、ロケーション・テーブルのエントリーを検索する。エントリーが検索できたら、そのエントリーが指しているブロックのアクセスを行う。このような順序でアクセスが実行されるので、排他はロケーション・テーブル→ブロックの順序となる。ロケーション・テーブルを排他するのは、ロケーション・テーブルのエントリーに対して更新が行われる可能性があるからである。

【０２８７】

プライマリー・ブロックにオーバーフロー・ブロックが連結されている場合には、オーバーフロー・ブロックも同時に排他対象とする。これは、レコードの追加・挿入の場合には、レコードの移動が行われて、オーバーフロー・ブロックに対するアクセスが発生する可能性があること、更新の場合でも、レコード長が変更になると、同様に、オーバーフロー・ブロックに対するアクセスが発生する可能性があること、からである。レコードの更新によって代替キーが変更された場合には、その代替キーに対する代替キー・テーブルの該当するエントリーを更新する。これは、代替キー・ロケーション・テーブルを使用しない形態の場合には、代替キー・テーブルの該当する代替キー・ブロック及び、その代替キー・ブロックに連結されている代替キー・オーバーフロー・ブロックを排他することになり、代替キー・ロケーション・テーブルを使用する形態の場合には、まず、代替キー・ロケーション・テーブルの排他を行い、次に該当する代替キー・ブロック及び、その代替キー・ブロックに連結されている代替キー・オーバーフロー・ブロックを排他することになる。

これは、ロケーション・テーブル→ブロック→（代替キー・ロケーション・テーブル）→代替キー・ブロック、という排他順序となる。

【０２８８】

[代替キーの場合の排他方法]

代替キーでのアクセスの場合は、次のようになる。

代替キー・ロケーション・テーブルを使用しない形態の場合、まず、代替キー・テーブルをバイナリー・サーチし、ターゲット・代替キー値を持つエントリー

が含まれる代替キー・ブロックを検索する。代替キー・ロケーション・テーブルを使用する形態の場合には、まず、代替キー・ロケーション・テーブルをバイナリー・サーチで検索し、ターゲット・キー値が含まれるエントリーを検索し、そのエントリーが指している代替キー・ブロック内の代替キー・ブロック・エントリーを検索する。ここまでの検索は、排他を掛けずに実行する。排他しないことによる問題は、何ら発生しない。
目的の代替キー・ブロック・エントリーが見つかったら、今度は、そのエントリーの情報により、ロケーション・テーブルの検索を行うことになる。このロケーション・テーブルの検索は、既述したように、代替キー・ブロック・エントリーの形式によって異なる。

【０２８９】

目的の主キー値を持つロケーション・テーブルのエントリーが検索できたら、そのロケーション・テーブル・エントリーを排他する。その後、そのロケーション・テーブル・エントリーが指しているブロック及びオーバーフロー・ブロックの排他を行い、ブロック内のレコード検索を行う。レコードが検出できて、レコードの更新が行われた場合に、代替キー値が変更になった場合は、主キー値で検索を行う場合と同様である。
このように排他を実行することで、排他順序は、ロケーション・テーブル→ブロック→（代替キー・ロケーション・テーブル）→代替キー・ブロック、となる。

【０２９０】

代替キー・テーブルの形式の項で述べたように、代替キーのエントリーに、ブロックのアドレスを保持している場合、代替キー・ブロックのエントリーが見つかると、そのエントリーにブロックのアドレスが保持されているため、ロケーション・テーブルを経ずにブロックへアクセスを行うことになる。このため、排他の順序が逆（ブロック→ロケーション・テーブル）になってしまう。このため、代替キー・テーブルのエントリーにブロック・アドレスを保持する方法はデッド・ロックの観点から好ましくない。

【０２９１】

[データ・バックアップ・リカバリー方式との連携]

本発明の再編成を、本発明者が提案した、「データ・バッ クアップ・リカバリー方式（特願2001-094678号）」に適用する場合の方式に関して図２０及び図２１を参照して説明する。

【０２９２】

図２０は、本第１ないし第３の発明が適用されるデータ・バッ クアップ・リカバリー方式において、同期密結合方式を採用した場合の動作を説明するためのフローチャートである。

図２１は、本第１ないし第３の発明が適用されるデータ・バッ クアップ・リカバリー方式において、非同期疎結合方式を採用した場合の動作を説明するためのフローチャートである。

この「データ・バックアップ・リカバリー方式」では、図２０及び図２１に示すように、データの検索・更新を行うためのプライマリー・システム１と、そのデータをバックアップしておくセカンダリー・システム２とから構成されている。なお、プライマリー・システム１には、図２０及び図２１に示すように、バックアップ制御機構１０４が設けられている。また、セカンダリー・システム２も、図１、図２、図９、あるいは、図１２に示すような、ロケーション・テーブル、代替キー・テーブル、あるいは、代替キー・ロケーション・テーブルを備えており、上述と同様の符号を用い、これにシステム番号を付加して説明することにする。プライマリー・システムのシステム番号を１、セカンダリー・システムのシステム番号を２とする。

【０２９３】

図２０に示す方式では、プライマリー・システム１のデータに変更が加えられるたびに、セカンダリー・システム２に対して変更通知を出し、セカンダリー・システム２のデータを変更する、という仕組みを採用している。

【０２９４】

ここで、上記方式において使われるログについて説明すると、データの変更内容を含むログが、Ａログである。トランザクション・キャンセル時に、データを元に戻すためのログが、Ｂログである。Ｂログは、データを一定の過去の状態に戻せるような目的も有している。トランザクションのデータがＴログである。プ

ログラム・エラーによって、データが誤った更新をされた場合に、Ｂログを用いてプログラムが異常になる直前までデータを復元し、正常なプログラムに入れ替えた後、Ｔログを使用して、データを正常化する、ということが可能となっている。また、セカンダリー・システム２は必要に応じて、１つ以上設けることが可能である。

上述した「データ・バックアップ・リカバリー方式」では、セカンダリー・システム２にＡログが送信されると、プライマリー・システム１と同期を取ってデータ更新を行う同期密結合方式と、セカンダリー・システム２では同期をとらず、遅れてデータの更新を行う非同期疎結合方式の２通りが考案されている。

【０２９５】

［同期密結合方式と非同期疎結合方式の採用のポイント］

同期密結合方式と非同期疎結合方式の採用のポイントについて説明する。周知のとおり、光の速度は、約３００，０００［ｋｍ／秒］である。これに基づいてデータ伝送時間について計算すると、以下のようになる。伝送距離が３００［ｍ］のときには１［μｓ］の伝送時間がかかり、伝送距離が３０［ｍ］のときには１００［ｎｓ］の伝送時間がかかり、さらに、３［ｍ］のときには１０［ｎｓ］伝送時間がかかることになる。

また、データの伝送速度に関しては、次の様な計算が成り立つ。１［Ｇビット／ｓ］の伝送スピードを持つ機器上で、１［Ｋバイト］のデータを伝送するための時間は、１［Ｇビット／ｓ］≒１００［Ｍバイト／ｓ］として計算すると、１０［μ秒］が必要となる。

同期密結合方式の場合には、距離に比例した伝送遅延と、データ量に比例した、伝送時間が必要となる。このため、これらの時間が必要なことを前提にして、方式を採用することが必要である。

【０２９６】

＜再編成をトランザクションとして扱う＞

前述したように、本再編成は、一時点で、１個から数十個のブロックや代替キー・ブロック等を対象として再編成を実施するのであり、この再編成は、見かけ上、通常のデータ処理のトランザクションとして扱うことになっている。

上述した国内優先「特願2001-094678「データ・バックアップ・リカバリー方式」では、トランザクション単位でバックアップを実施し、リカバリーに関しても、トランザクションが完了しているか否かによって、リカバリーの範囲を決定している。

本再編成は、一つのトランザクションとして実施することにより、上記「データ・バックアップ・リカバリー方式」との矛盾が生じないよう、工夫されている。

【０２９７】

[同期密結合方式に有利なＡログ再編成方式]

まず、同期密結合方式の場合に有効な、Ａログ再編成方式関して説明を行う。本発明の再編成を実施する場合には、プライマリー・システム１とセカンダリー・システム２が、同時に再編成を実施する。例えば、プライマリー・システム１で、ロケーション・テーブルのエントリー「１２３」に対して再編成を実行する場合には、セカンダリー・システム２の総てで、セカンダリー・ロケーション・テーブルのエントリー「１２３」の再編成を実施する。この場合に、セカンダリー・システム２は、プライマリー・システム１から送信されるＡログを用いてセカンダリー・システム２のデータ更新を行う方法である。

【０２９８】

まず、本第１の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムを「データ・バックアップ・リカバリー方式」に適用し、ロケーション・テーブルとブロックの再編成の場合に関して説明する。

再編成の方式は既に詳細に説明してあり、これはセカンダリー・システム２でも同様であるので簡単な説明とし、プライマリー・システム１とセカンダリー・システム２の連携に関する部分を重点的に説明する。本説明では、セカンダリー・システムが１台の場合に関して説明を行うが、複数台の場合も同様となる。セカンダリー・システムのシステム番号を「2」とする。

【０２９９】

次に、以前の説明で用いた、図３と図２０を使用して以下にプライマリー・システムとセカンダリー・システムの動作を説明する。

この図３及び図２０において、再編成を実施するためには、その旨をプライマリー・システム１からセカンダリー・システム２に対して通知するログが必要となる。これをＲＳ情報とする。これは、再編成をスタートするという情報であり、どのキー（ロケーション・テーブルまたは代替キー・テーブル）に対する再編成であるかの情報の他に、適正な初期格納率等の情報を含んでいる。この場合の再編成対象は、ロケーション・テーブルである。

【０３００】

図２０では、再編成トランザクション［1］の処理に入り（図２０のＳ３０１）、プライマリー・システム１からセカンダリー・システム２に対して、最初にＲＳ情報の送信を行っている（図２０のＳ３０２）。

セカンダリー・システム２では、ＲＳ情報を受信後（図２０のＳ４０１）は、再編成ポインターＲＰＬＣ２、ＲＰＬＮ２の作成と、新規ロケーション・テーブルＬＮ２の領域を確保する（図２０のＳ４０２）。再編成ポインターＲＰＬＣ２の初期値は新規ロケーション・テーブルＬＣ２の先頭アドレス、新規再編成ポインターＲＰＬＮ２の初期値は新規ロケーション・テーブルＬＮ２の先頭アドレスである。それらが完了したら、セカンダリー・システム２は、ＲＳ－ＡＣＫ２情報をプライマリー・システム１に対して送信する（図２０のＳ４０３）。

プライマリー・システム１では、ＲＳ－ＡＣＫ２情報を受信した後に（図２０のＳ３０３）、具体的な再編作業を開始することになる。

以上の一連の動作は、プライマリー・システム１及びセカンダリー・システム２ではトランザクションとする。図２０では、再編成トランザクション［1］として、通常のトランザクションと区別できるようにしているが、実際にトランザクションの種類を分ける必要は無く、通常のトランザクションと同様に扱うことが可能である。

【０３０１】

図２０では、プライマリー・システム１において、次に通常のトランザクション［2］の処理が実行されている（図２０のＳ３０４）。次に、プライマリー・システム１において、再編成トランザクション［3］が実行される（図２０のＳ３０５）。プライマリー・システム１では、図３に示すように、ロケーション・

テーブルＬＣ１，ＬＮ１と、必要に応じてブロック１０を参照し、１回目の再編成の範囲を決定する。例えば、ブロック１０のブロック番号「０」、ブロック１０のブロック「１」とそのオーバーフロー・ブロック１３のブロック番号「１－２」、そのオーバーフロー・ブロック１４のブロック番号「１－３」が対象となったとする。この場合、ＲＳＥＳ情報で、ブロック１０のブロック番号「０」、「１」、オーバーフロー・ブロック１３，１４のブロック番号「１－２」、「１－３」を排他すること、それらのブロックを１回目の再編成の対象として再編成を実行することをプライマリー・システム１からセカンダリー・システム２に通知し（図２０のＳ３０６）、プライマリー・システム１でブロック１０のブロック番号「０」、「１」、オーバーフロー・ブロック１３，１４のブロック番号「１－２」，「１－３」の排他を実行する。

【０３０２】

セカンダリー・システム２では、ＲＳＥＳを受信したら（図２０のＳ４０４）、再編成が実行できないような特別の場合を除いて、直ちに、ブロックのブロック番号「０」、「１」、オーバーフロー・ブロックのブロック番号「１－２」、「１－３」の排他を実行する（図２０のＳ４０５）。その後、セカンダリー・システム２は、排他完了通知であるＲＳＥＳ－ＡＣＫ２情報をプライマリー・システム１に送信する（図２０の４０６）。

プライマリー・システム１では、その後、ブロック１０のブロック番号「０」、「１」、「１－２」、「１－３」の再編成を実行する。その結果として、ロケーション・テーブルＬＮのエントリー、及び、ブロック１０のブロック番号「０」、「１」、オーバーフロー・ブロック１３，１４のブロック番号「１－２」、「１－３」に変更が発生するが、その変更をＡログとしてセカンダリー・システム２に送信する（図２０のＳ３０７～Ｓ３１１）。これは、通常のレコード更新に伴う、Ａログと論理的には同じものであるので、セカンダリー・システム２では、Ａログを受信した後、そのＡログを該当する、エントリーとブロックに適用する（図２０のＳ４０７～Ｓ４１０）。

【０３０３】

Ａログの適用に当たっては、次の様にする。すなわち、Ａログには、ロケーシ

ョン・テーブルのどのエントリーのものか、どのブロックのものかを識別するために、ブロック番号を通知する。ブロック番号の通知は、プライマリー・ブロックに対して、オーバーフロー・ブロックが連結されている場合に、それらを一体として作業する場合には、プライマリー・ブロックと、それに連結されているオーバーフロー・ブロック全体の更新後の情報をＡログとして送信する方法を採用することが可能であるが、この場合には、送信するデータ量が多くなるため、何番目のオーバーフロー・ブロックに対するログであるかの識別番号を併せて送信することで、送信データ量を削減することができる。この他、プライマリー・ブロックとそれに連結されているオーバーフロー・ブロック全体を一体としてみなした上で、更新があった部分のみを抜き出して、変位と長さ、更新後データという形式で送信することも可能である。

【０３０４】

＜ロジック送信方式＞

更に、次の方式も有効である。再編成の場合には、ブロック全体としては変化があるが、レコードの内容は変化しない。特に、フラグメンテーションの解消や、適正な初期格納率の確保を行う場合には、レコードをブロック内やブロック間で移動することが多くなるが、この場合、移動するロジックそのものを、Ｌ（論理）ログとして送信することにより、送信データ量を大きく削減できる。ロジックの例としては、オーバーフロー・ブロック内の全レコードは、１５００バイト右に移動し、その後、プライマリー・ブロック内の９１番目以降のレコードを、オーバーフロー・ブロックに移動する、といったものである。

【０３０５】

セカンダリー・システム２では、このＬログに基づき、当該ブロックに対して、その論理を適用して操作を行う（図２０のＳ４０５）。

送信量について考察すると、ブロックの大きさが１６［Kバイト］であったとして、プライマリー ・ブロック１の内部には１５０［バイト］のレコードが１００［個］入っており、オーバーフロー・ブロックには同じく１５０［バイト］のレコードが８０［個］入っていた場合、Ａログで更新後情報をそのまま送信すると、１５０×９０＝１３５００［バイト］の送信の他に、ブロックの情報の送

信が必要となる。

【０３０６】

一方、ロジックの送信では、概ね、１０００［バイト］以内の送信量で済むと考えられる。また、ロジックはコンパイルが必要なものは実行が面倒になるため、インタープリターなどですぐに実行できる形式のものが好ましい。
また、当然のことながら、このロジックは、その時点の再編成分に適用するだけのもので、１回の適用が完了したら廃棄する。

【０３０７】

＜データ・バックアップ・リカバリー方式にロジック転送を適用＞

このロジックを転送する方式は、上記「データ・バックアップ・リカバリー方式」に適用することも可能である。データ・バックアップ・リカバリー方式では、変更後のブロックの内容そのもの、または、変更分をセカンダリー・システムに送信する方式を採用しているが、レコードが挿入された場合には、複数のレコードがブロック内で移動することになる。レコード自体には変更は無いため、Ｌログの送信を行い、セカンダリー・システムで、送信されたロジックに基づき、当該ブロック内のレコードの操作を行う。

【０３０８】

また、ロケーション・テーブルとブロックの再編成の項で述べたように、ブロックの再編成を行うと、代替キー・テーブルのエントリーの形式によって、代替キー・テーブルに対する更新が発生する場合があるので、それも同様にＡログとして送信する。

さて、再編成を行っている場合には、Ａログとして送信される情報が、現用の新規ロケーション・テーブルＬＣ１のものであるか、新規ロケーション・テーブルＬＮ２のものであるかを識別する必要がある。そうしないと、誤った更新を行ってしまうからである。これを防止するためには、Ａログ中に、対象が現用ロケーション・テーブルＬＣ２であるか、新規ロケーション・テーブルＬＮ２であるかの識別を保持する。これにより、誤った更新を防止できる。また、Ａログには、再編成ポインターを含める必要がある。再編成ポインターは、再編成の進捗を知る上で重要であるが、セカンダリー・システム２を参照用のシステムとして使

用している場合には、再編成ポインターが無い場合は動作ができなくなってしまうからである。

【０３０９】

プライマリー・システム１では、ブロック１０のブロック番号「０」、「１」、「１－２」、「１－３」の再編成を行い（図２０のＳ３０４）、ブロックが完成する都度、ブロックとロケーション・テーブルの変更後の内容をＡログとして、セカンダリー・システムに送信する（図２０のＳ３０８～Ｓ３３１１）。セカンダリー・システム２では、Ａログを受信すると（図２０のＳ４０７～Ｓ３４１０）、その情報に従って、新規ロケーション・テーブルＬＮ２とブロックを書き換える（図２０のＳ４０５）。

【０３１０】

プライマリー・システム１で、ブロック１０のブロック番号「０」、「１」、オーバーフロー・ブロック１３，１４のブロック番号「１－２」、「１－３」の再編成が完了したら、当該ブロックの排他を解除し、ＲＳＥＥ情報をセカンダリー・システムに送信する（図２０のＳ３１２）。セカンダリー・システム２では、ＲＳＥＥ情報を受信し（図２０のＳ４１１）、かつ、ブロックのブロック番号「０」、「１」、オーバーフロー・ブロックのブロック番号「１－２」、「１－３」に対する処理が完了した後、当該ブロックの排他を解除し、ＲＳＥＥ－ＡＣＫ２情報をプライマリー・システム１に送信する（図２０のＳ４１２）。

このようにして、順次、プライマリー・システム１で再編成を行いながら、Ａログをセカンダリー・システム２に送信し、セカンダリー・システム２では直ちに該当するブロック等の更新を行うことにより、プライマリー・システム１とセカンダリー・システム２を同期しながら、同時に再編成を行うことが可能となる。

順次、再編成を実行し、現用ロケーション・テーブルＬＣのラスト・ポインターが指すエントリーの直前まで再編成が完了したら、ロケーション・テーブルとブロック全体の再編成が完了したことになる。

【０３１１】

次に、本第２又は第３の発明を「データ・バックアップリカバリー方式」に適

用した場合について説明する。

まず、代替キー・テーブルの再編成について図２１を参照して説明する。すなわち、代替キー・テーブルの再編成であるが、ロケーション・テーブルとブロックの再編成で述べたように、一時点で再編成の対象とする代替キー・ブロックと代替キー・オーバーフロー・ブロックを決定し、その情報をプライマリー・システム１からセカンダリー・システム２に送信する。その後、再編成によって更新されたデータを、Ａログとして、プライマリー・システム１からセカンダリー・システム２に送信し、セカンダリー・システム２では、その情報に基づき該当する代替キー・ブロックと、代替キー・ロケーション・テーブルを使用する場合には、代替キー・ロケーション・テーブルの更新を行う。

【０３１２】

代替キー・テーブルの再編の場合にも、ＡログではなくＬログを送信することにより、伝送量を削減することが可能である。

[非同期疎結合方式に有利な併行再編成方式の場合]

次に非同期疎結合方式の場合に関して説明する。非同期疎結合方式の場合にもＡログ更新方式を使用することは可能であるが、実際には伝送路の距離による遅延が発生する他、伝送路の容量の大きさによってもＡログ送信の遅延が発生する可能性があるために、セカンダリー・システムでの更新が遅延する可能性が大きい。

【０３１３】

非同期疎結合方式の場合に有効な、併行再編成方式関して説明を、図２１を参照して行う。本発明の再編成を実施する場合には、図２１に示すように、プライマリー・システム１とセカンダリー・システム２が、同時に再編成を実施する。例えば、プライマリー・システム１で、ロケーション・テーブルのエントリー「１２３」に対して再編成を実行する場合には、セカンダリー・システム２の総てで、セカンダリー・ロケーション・テーブルのエントリー「１２３」の再編成を実施する。つまり、プライマリー・システム１とセカンダリー・システム２が個別でありながら全く同一の動作を行う。これは、セカンダリー・システム２が複数ある場合には、そのすべてのセカンダリー・システム２が、プライマリー・シ

ステム１と同時に再編成を実行する、ということである。

【０３１４】

再編成を実施するためには、その旨をプライマリー・システム１からセカンダリー・システム２に対して通知するログが必要となる。これをＲＳ情報とする。このＲＳ情報は、再編成をスタートするという情報であり、どのキー（ロケーション・テーブルまたは代替キー・テーブル）に対する再編成であるかの情報の他に、適正な初期格納率等の情報を含んでいる。この場合の再編成対象は、ロケーション・テーブルである。プライマリー・システム１は、このＲＳ情報をセカンダリー・システム２に送信する（図２１のＳ５０１）。

【０３１５】

セカンダリー・システム２では、ＲＳ情報を受信後（図２１のＳ６０１）は、再編成ポインターＲＰＬ２、ＲＰＬＮ２の作成と、新規ロケーション・テーブルＬＮ２の領域を確保する（図２１のＳ６０２）。セカンダリー・システム２では、それらが完了したら、ＲＳ－ＡＣＫ２情報をプライマリー・システム１に対して送信する（図２１のＳ６０３）。

プライマリー・システム１では、ＲＳ－ＡＣＫ２情報を受信した後に（図２１のＳ５０２）、具体的な再編作業を開始することになる。

【０３１６】

プライマリー・システム１では、ロケーション・テーブルと、必要に応じてブロックを参照し、１回目の再編成の範囲を決定する。プライマリー・システム１では、例えば、ブロックのブロック番号「０」、「１」と、そのオーバーフロー・ブロックのブロック番号「１－２」、「１－３」が対象となったとする。この場合、ＲＳＥＳログで、ブロックのブロック番号「０」、「１」と、そのオーバーフロー・ブロックのブロック番号「１－２」、「１－３」を排他すること、それらのブロックを１回目の再編成の対象として再編成を実行することを、ＲＳＥＳ情報によりプライマリー・システム１からセカンダリー・システム２に通知する（図２１のＳ５０３）。

【０３１７】

セカンダリー・システム２では、ＲＳＥＳを受信したら（図２１のＳ６０４）

、再編成が実行できないような特別の場合を除いて、ＲＳＥＳ－ＡＣＫ２をプライマリー・システムに送信する（図２１のＳ６０５）。その後（図２１のＳ６０５）、セカンダリー・システム２は、オーバーフロー・ブロックの解消、フラグメンテーションの解消、適正な初期格納率の確保を、プライマリー・システムと同じロジックで実行する（図２１のＳ６０６）。また、プライマリー・システム１は、排他を解除したときには、排他解除通知としてＲＳＥＥ情報をセカンダリー・システム２に送信する（図２１のＳ５０４）。

【０３１８】

セカンダリー・システム２では、ブロックのブロック番号「０」、「１」、ブロック番号「１」に連結しているオーバーフロー・ブロックのブロック番号「１－２」、「１－３」の再編成をし（図２１のＳ６０６）、それが完了したら、排他解除完了通知としてＲＳＥＥ－ＡＣＫ２情報をプライマリー・システム１へ送信する（図２１のＳ６０７）。

プライマリー・システム１とセカンダリー・システム２は同期していないので、セカンダリー・システム２で、ブロックのブロック番号「０」、「１」、そのブロック番号「１」に連結されているオーバーフロー・ブロックのブロック番号「１－２」、「１－３」の再編成が完了していなくとも、次のブロックに対する再編成の指示をセカンダリー・システムに送信することができる。

【０３１９】

次に、代替キー・テーブルの再編成について説明する。すなわち、代替キー・テーブルの再編成であるが、ロケーション・テーブルとブロックの再編成で述べたように、一時点で再編成の対象とする代替キー・ブロックと代替キー・オーバーフロー・ブロックを決定し、その情報をプライマリー・システムからセカンダリー・システムに送信する。その後、セカンダリー・システムでは再編成を実施し、一時点で再編成の対象となった代替キー・ブロックの再編成が完了したら、セカンダリー・システムからプライマリー・システムに対してＲＳＥＥ情報を送信する。

【０３２０】

［リカバリー方式］

リカバリーは、同期密結合方式のセカンダリー・システムから実行することが好ましい。何故ならば、非同期疎結合方式の場合は、距離が離れており、伝送路に十分な容量が無いためにリカバリーに時間が掛かる可能性があるからである。リカバリーの要求が発生した場合は、まず、現在のトランザクションがプライマリー・システムで完了しているか否かを確認する。プライマリー・システムで完了している場合には、セカンダリー・システムでもそのバックアップ・トランザクションを完了させる。プライマリー・システムでトランザクションションが完了しておらず、キャンセルされた場合は、セカンダリー・システムでは、当該バックアップ・トランザクションに関係するＢログを用いて、セカンダリー・システムのデータを、トランザクションの実行前に戻す。

【０３２１】

その後、プライマリー・システムから要求された、ロケーション・テーブルのエントリー、ブロック（プライマリー・ブロックとオーバーフロー・ブロック）、代替キー・ロケーション・テーブルのエントリー、代替キー・ブロック、代替キー・オーバーフロー・ブロックの何れか、または、すべてを、セカンダリー・システムからプライマリー・システムへ送信する。

【０３２２】

プライマリー・システムでは、セカンダリー・システムから送信されてきた情報に基づいてデータの修復を行う。データの修復は、ハードウエアのトラブルの場合と、単なる情報の欠如の場合で異なる。

ハードウエアのトラブルの場合は、プライマリー・システムでは元のデータがあった場所に情報を書き戻すことが不可能であるので、新たな格納領域にリカバリーすべき対象の領域を確保する。例えば、ロケーション・テーブルの一部のエントリーの格納領域が破壊された場合には、その部分だけを新たに確保して、不連続ロケーション・テーブルとして使用することも可能であるし、ロケーション・テーブル全体に対して新たに領域を確保し、連続領域のロケーション・テーブルとすることも可能である。ロケーション・テーブルは、ブロックのアドレスを保有しているが、ブロック自体がリカバリーの対象になっていない場合は、エントリー情報の変更は発生しない。

【０３２３】

ブロックの場合は、当該ブロックに対する領域を確保して、セカンダリー・システムから送信されたリカバリー情報に基づいて、ブロックに書き込んでいけばよい。但し、この場合は、該当ブロックのアドレスが変更されるため、ロケーション・テーブルの該当エントリーのブロック・アドレスを書き換える必要がある。更に、代替キー・エントリーにブロック・アドレスを保有する場合には、該当する代替キー・エントリーの書き換えが必要となり、リカバリーの時間が掛かることになる。このため、代替キー・エントリーの形式を選択するには、このような面からの検討も必要である。

【０３２４】

〔付記〕

なお、データ・データしステムとしては、次のようにしてもよい。すなわち、コンピューターにおけるデータおよびデータベース・システムにおいて、代替キーと該当レコードが格納されているブロック番号と該当レコードの主キーとからなるエントリーを複数格納可能とするとともに、予め同一大きさで必要数を連続して確保できる代替キー・ブロックを使用し、前記代替キー・ブロックの記憶装置内での位置を管理するため、前記代替キー・ブロックに付与した番号と前記記憶装置内の物理的位置を対応させた代替キー・ロケーション・テーブルとを用い、前記キー・ブロックに代替キー･エントリーを代替キーの順番で格納するとともに、前記代替キー・ブロックに格納できなくなったときに、新たな代替キー・オーバーフロー・ブロックを割り当てて代替キー･エントリーを格納し、かつ、前記代替キー・ロケーション・テーブルにより前記代替キー・ブロックの前記記憶装置内での位置管理を行うことを特徴とするデータおよびデータベース・システムであって、データを代替キーで検索するときは、この代替キー・ロケーション・テーブルを検索することにより、目的の代替キー値を持つエントリーを検出し、そのエントリーから目的のレコードが格納されているブロックを知り、そのブロック中の当該レコードの検索を行うことができる。

【０３２５】

【発明の効果】

請求項１記載の発明によれば、現用のロケーション・テーブルと新規のロケーション・テーブルを用意し、単位処理時点で１個又は複数のブロックを対象とし、時間現用のロケーション・テーブルのエントリーを新規のロケーション・テーブルに順次書き移してゆくので、システムの運用を停止することなく再編成を行うことができ、かつ、従来のシステムに比較して大幅な記憶領域の省略ができる。

【０３２６】

請求項２記載の発明によれば、ブロック内の格納率が所定の値の範囲以外である場合に前後のブロックのレコードを移動してフラグメンテーションを解消するようにしたので、ブロック内を適正な格納率に維持できる。

【０３２７】

請求項３記載の発明によれば、現用のロケーション・テーブル及び新規のロケーション・テーブルにそれぞれ再編成ポインターを設け、各再編成ポインターに対して、単位処理時点で一個または複数個のブロックを対象として順次前記再編成処理が終了した位置を格納し、再編成がラスト・ポインターまで到達したときに再編成処理を完了するので、システムの運用を停止することなく再編成を行うことができ、かつ、従来のシステムに比較して大幅な記憶領域の省略ができる。

【０３２８】

請求項４記載の発明によれば、再編成をしている最中に、レコードを主キーで検索する場合には、目的の主キー値が再編成ポインターの指しているエントリーのプライマリー・ブロック及びオーバーフロー・ブロックに含まれるレコードの主キー値より大きいか小さいかを比較し、目的の主キー値が、再編成ポインターが指しているブロックに格納されているレコードの主キー値以上と判断されたときには現用のロケーション・テーブルを使用して目的のレコードを検索し、主キー値未満と判断されたときには新規のロケーション・テーブルを使用して目的のレコードを検索するので、再編成中でも目的のレコードを確実に検索することができる。

【０３２９】

請求項５記載の発明によれば、データベース再編成指令を受けると、前記現用

代替キー・テーブルに加えて現用代替キー・テーブルを作成し、単位処理時点で１個または複数個のブロックを対象とし、現用代替キー・テーブルのエントリーを現用代替キー・テーブルに順次、書き移してゆき、書き移してゆく際に、代替キー・ブロックに連結されている代替キー・オーバーフロー・ブロックを検出したときには、当該代替キー・オーバーフロー・ブロックの連結を切り離し、新たな代替キー・ブロックとして現用代替キー・テーブルに書き移すので、システムの運用を停止することなく再編成を行うことができ、かつ、従来のシステムに比較して大幅な記憶領域の省略ができる。

【０３３０】

請求項６記載の発明によれば、前記代替キー・ブロック内の格納率が所定の値の範囲外である場合に前後の代替キー・ブロックのレコードを移動してフラグメンテーションを解消するので、代替キー・ブロック内の格納率を適正なものにすることができる。

【０３３１】

請求項７記載の発明によれば、現用代替キー・テーブル及び現用代替キー・テーブルにそれぞれ再編成ポインターを設け、各再編成ポインターに対して、単位処理時点で一個または複数個のブロックを対象として順次前記再編成処理が終了した位置を格納し、再編成がラスト・ポインターまで到達したときに再編成処理を完了するので、システムの運用を停止することなく再編成を行うことができ、かつ、従来のシステムに比較して大幅な記憶領域の省略ができる。

【０３３２】

請求項８記載の発明によれば、再編成をしている最中に、レコードを、代替キーで検索する場合に、目的の代替キー値が再編成ポインターの指しているエントリーの代替キー・ブロックに含まれるエントリーの代替キー値より大きいか小さいかを比較し、目的の代替キー値が再編成ポインターの指している代替キー・ブロックに格納されているエントリーの代替キー値以上と判断されたときには現用代替キー・テーブルを使用して目的のエントリーを検索し、前記エントリーの代替キー値未満と判断されたときには現用代替キー・テーブルを使用して目的のレコードを検索するので、再編成中でも目的のレコードを確実に検索することがで

きる。

【０３３３】

請求項９記載の発明によれば、データベース再編成指令を受けると、前記現用代替キー・ロケーション・テーブルに対して新規に代替キー・ロケーション・テーブルを作成し、単位処理時点で1個または複数のブロックを対象とし、現用代替キー・ロケー ション・テーブルのエントリーを新規の代替キー・ロケーション・テーブルに順次、書き移してゆき、順次、書き移してゆく際に、代替キー・ブロックに連結されている代替キー・オーバーフロー・ブロックを検出したときには、当該代替キー・オーバーフロー・ブロックの連結を切り離し、新規の代替キー・ロケーション・テーブルに新しいエントリーを追加し、新規の代替キー・ロケーション・テーブルの代替キー・ブロックとするので、システムの運用を停止することなく再編成を行うことができ、かつ、従来のシステムに比較して大幅な記憶領域の省略ができる。

【０３３４】

請求項１０記載の発明によれば、前記代替キー・ブロック内の格納率が所定の値の範囲外である場合に前後の代替キー・ブロックのレコードを移動してフラグメンテーションを解消するので、代替キー・ブロック内の格納率を適正なものにすることができる。

【０３３５】

請求項１１記載の発明によれば、現用代替キー・ロケーション・テーブル及び新規の代替キー・ロケーション・テーブルにそれぞれ再編成ポインターを設け、前記各再編成ポインターに単位処理時点での再編成が終わっている位置を格納するので、システムの運用を停止することなく再編成を行うことができ、かつ、従来のシステムに比較して大幅な記憶領域の省略ができる。

【０３３６】

請求項１２記載の発明によれば、再編成をしている最中に、レコードを、再編成を行っている代替キーで 検索する場合には、目的の代替キー値が再編成ポインターの指しているエントリーの代替キー・ブロックに含まれるエントリーの代替キー値より大きいか小さいかを比較し、目的の代替キー値が再編成ポインター

の指している代替キー・ブロックに格納されているエントリーの代替キー値以上と判断されたときには現用代替キー・ロケーション・テーブルを使用して目的のレコードを検索し、前記エントリーの代替キー値未満と判断されたときには新規の代替キー・ロケーション・テーブルを使用して目的のレコードを検索するので、再編成中でも目的のレコードを確実に検索することができる。

【０３３７】

請求項１３記載の発明によれば、再編成を行うときに現用の代替キー・ロケーション・テーブルと新規の代替キー・ロケーション・テーブルを用意するだけでよいので、再編成に必要な領域が少なくて済む、さらに、再編成を行うときに代替キーテーブルの書き移しが最小限で済むので再編成に必要な時間が少なくて済む。

【図面の簡単な説明】

【図１】

本第１の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムが適用されるプライマリー・システムの一例を示す構成図である。

【図２】

図１で示したプライマリー・システムの内、本第１の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムが適用される部分のみを示した概略図である。

【図３】

本第１の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムの動作を説明するための図である。

【図４】

本第１の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいて、フラグメンテーションを解消する再編成の動作を説明するための図である。

【図５】

本第１の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成

システムにおいて、全体的なフラグメンテーションを解消させる方法について説明するための図である。

【図６】

本第１の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいて、全体的なフラグメンテーションを解消させる方法について説明するための図である。

【図７】

本第１の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいて、再編成中のデータの検索・読み書き動作について説明するための図である。

【図８】

本第１の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいて、検索中に再編成が進行した場合への対処する動作について説明するための図である。

【図９】

本第２の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいて、再編成を説明するための図である。

【図１０】

本第２の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいて、フラグメンテーションを解消するための動作を説明するための図である。

【図１１】

本第２の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいて、検索中に再編成が進行した場合への対処について説明するための図である。

【図１２】

本第３の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムを説明するための図である。

【図１３】

本第３の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいて、プライマリー・システムの代替キー・テーブルの一つを説明するために示す図である。

【図１４】

本第３の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムの再編成の方法を説明するための図である。

【図１５】

本第３の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいて、フラグメンテーションを解消する方法を説明するための図である。

【図１６】

本第３の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいて、全体的なフラグメンテーションを解消する方法について説明するための図である。

【図１７】

本第３の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいて、ブロックの再利用を説明するための図である。

【図１８】

本第３の発明の実施の形態に係るデータ及びデータベースの無停止自動再編成システムにおいて、代替キーの検索を行っている最中に再編成が進んでしまい、検索開始時の現用再編成ポインターと検索終了時の現用再編成ポインターの位置が異なっている場合の動作についての説明図である。

【図１９】

本第１又は第３の発明に係るデータ及びデータベースの無停止自動再編成システムにおいて、ロケーション・テーブルを排他する場合の例を示す図である。

【図２０】

本第１ないし第３の発明が適用されるデータ・バッ クアップ・リカバリー方式において、同期密結合方式を採用した場合の動作を説明するためのフローチャートである。

【図２１】

本第１ないし第３の発明が適用されるデータ・バッ クアップ・リカバリー方式において、非同期疎結合方式を採用した場合の動作を説明するためのフローチャートである。

【符号の説明】

１　プライマリー・システム

２　セカンダリー・システム

１０　ブロック

１１　代替キー・テーブル

１２　プライマリー・ブロック

１３，１４　オーバーフロー・ブロック

１５，１６　代替キー・オーバーフロー・ブロック

１７　代替キー・ブロック

ＬＣ　現用ロケーション・テーブル

ＬＮ　新規ロケーション・テーブル

ＡＡＣ　現用代替キー・テーブル

ＡＡＮ　新規代替キー・テーブル

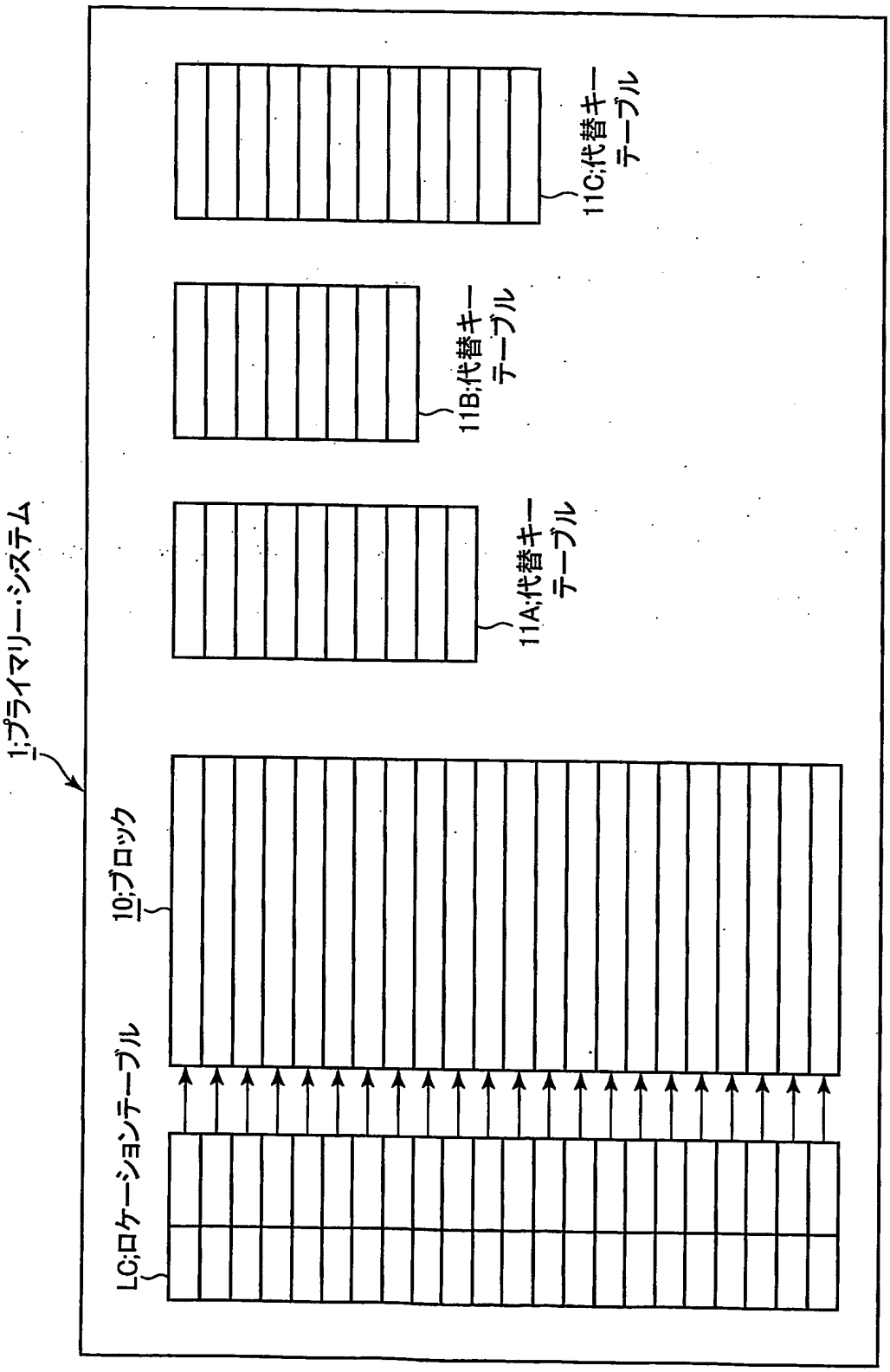
ＡＡＬＣ　現用代替キー・ロケーション・テーブル

ＡＡＬＮ　新規代替キー・ロケーション・テーブル

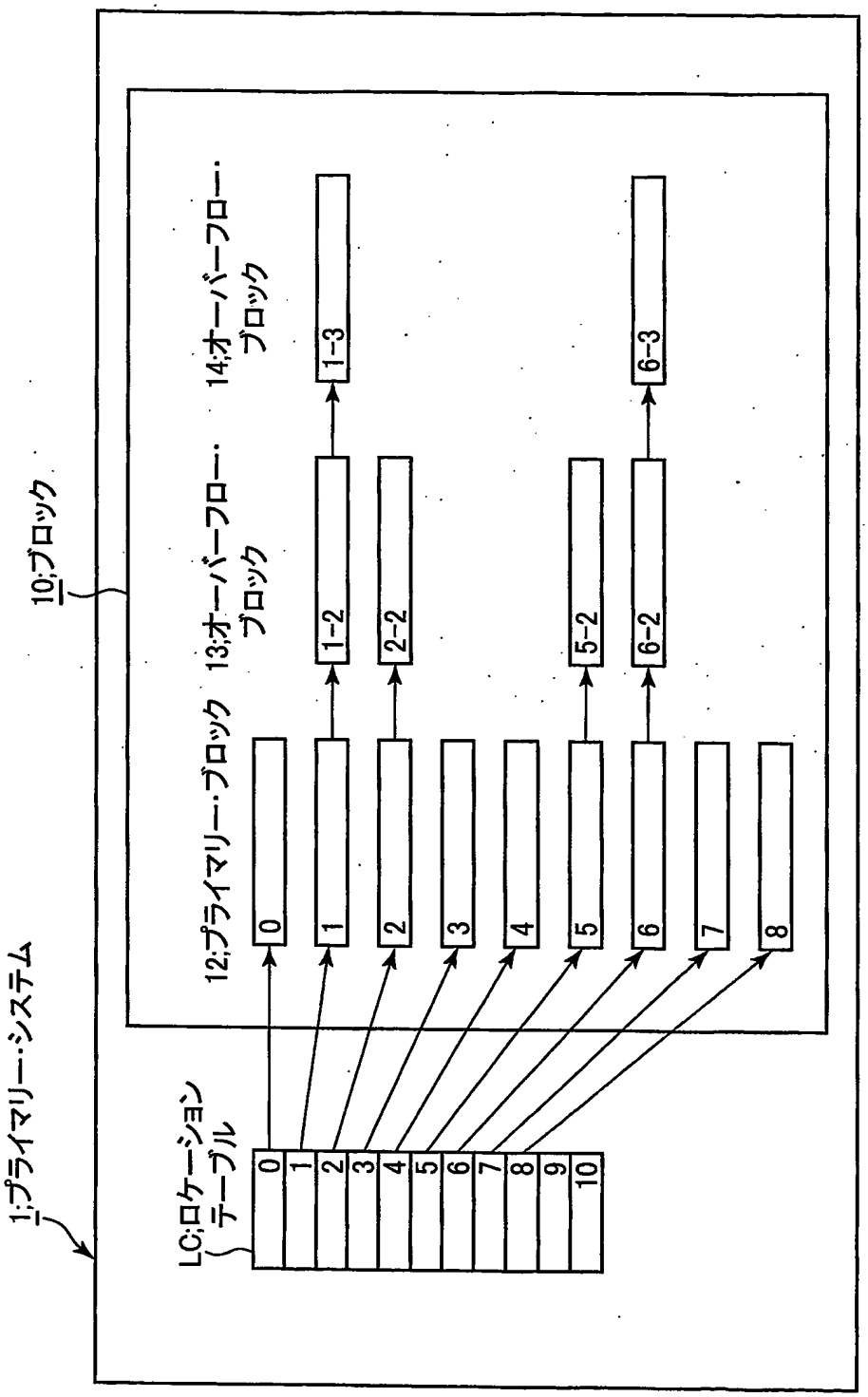
ＵＢＡＴ，ＵＡＢＡＴ　未使用ブロック・アロケーション・テーブル

【書類名】　図面

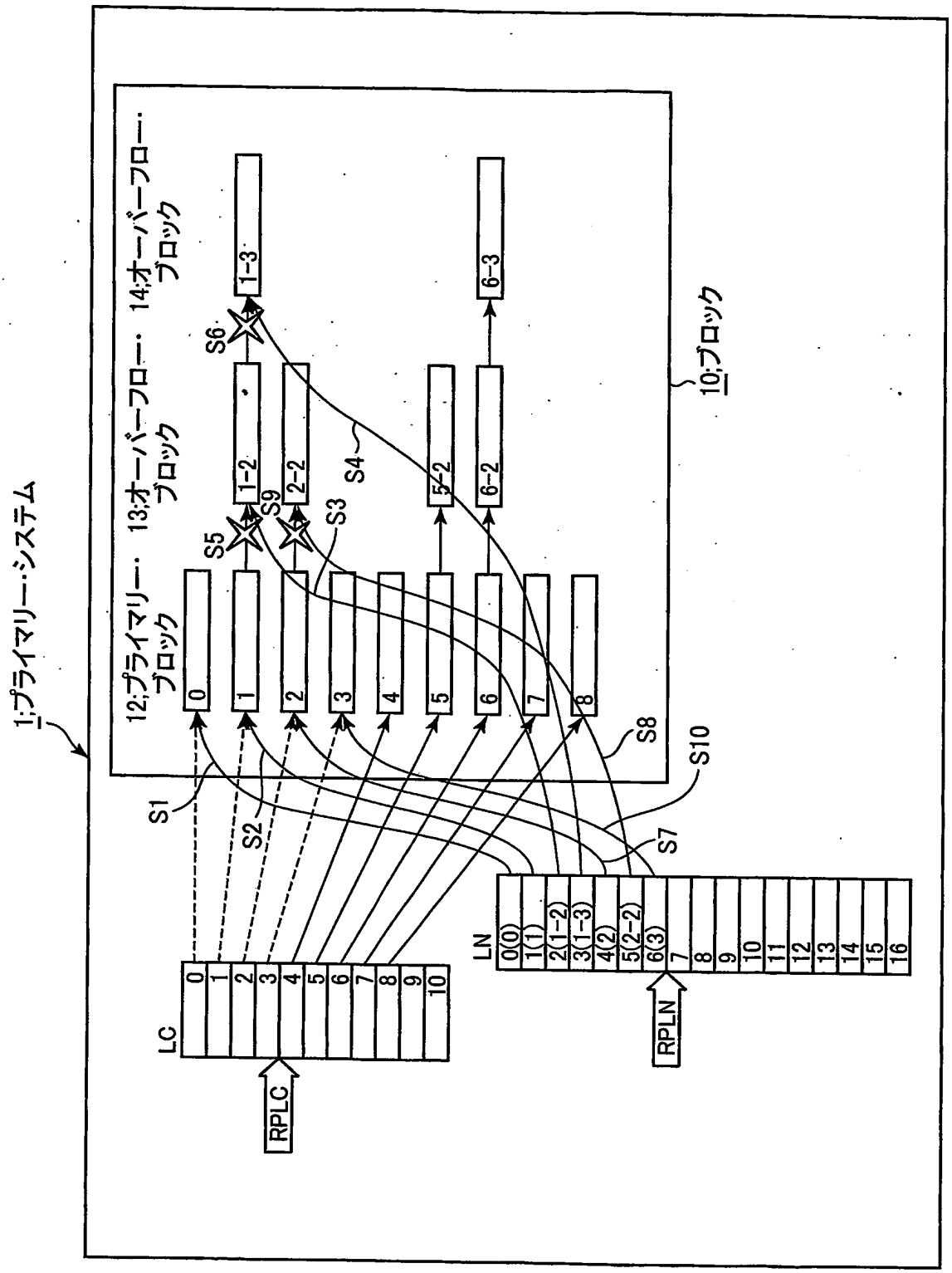
【図１】


1;プライマリー・システム
10;ブロック
LC;ロケーションテーブル
11A;代替キーテーブル
11B;代替キーテーブル
11C;代替キーテーブル

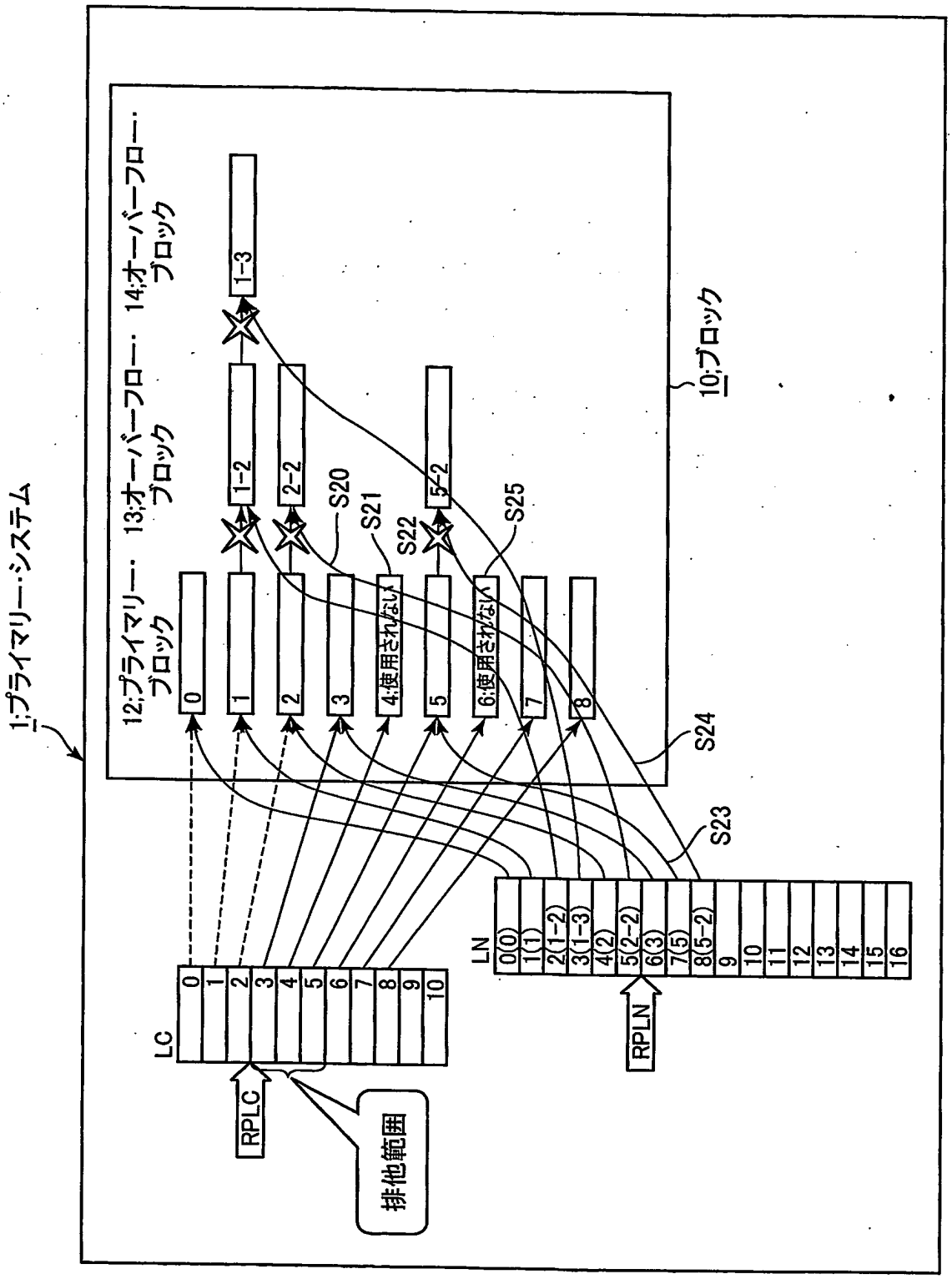
【図２】


1;プライマリー・システム
10;ブロック
LC;ロケーション テーブル
0
1
2
3
4
5
6
7
8
9
10
12;プライマリー・ブロック
13;オーバーフロー・ブロック
14;オーバーフロー・ブロック
1-2
2-2
5-2
6-2
1-3
6-3

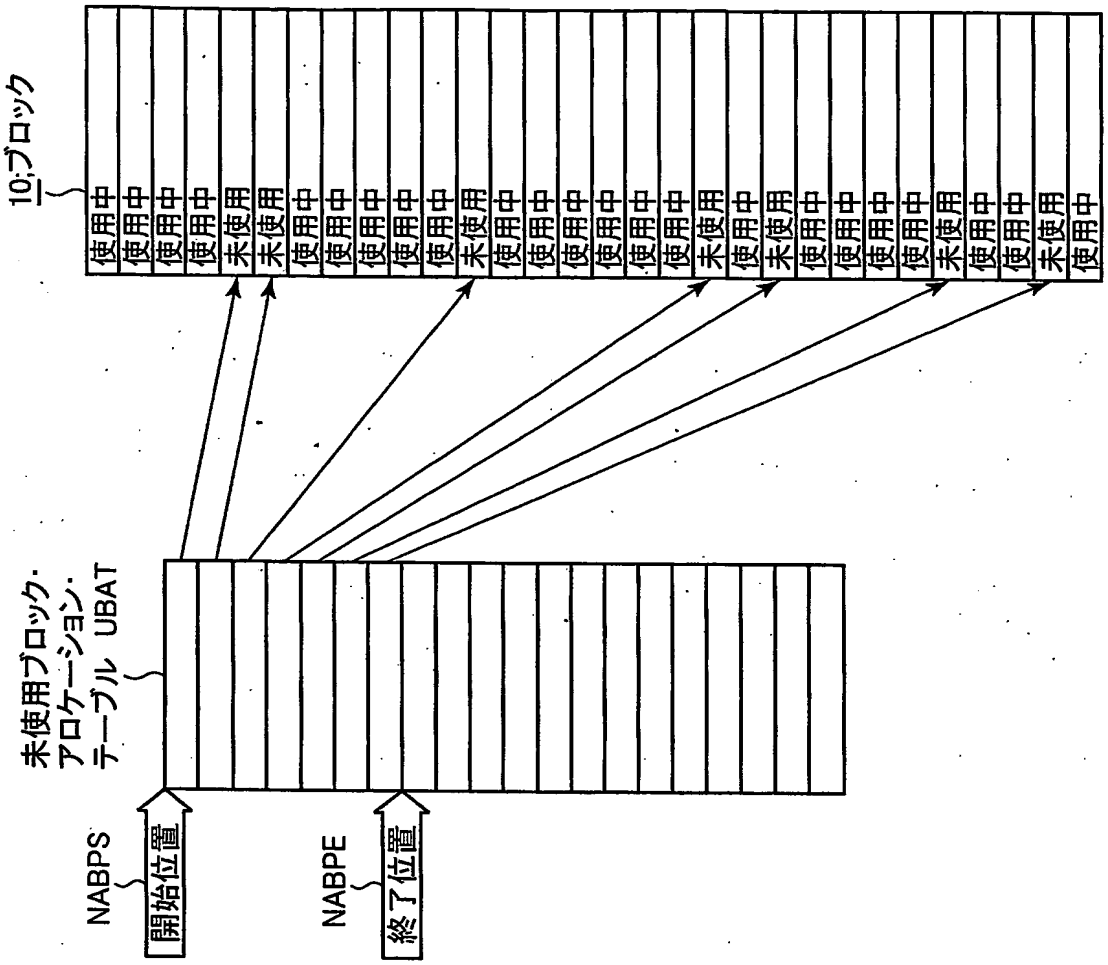
【図３】


1;プライマリー・システム
10;ブロック
12;プライマリー・ブロック
13;オーバーフロー・ブロック
14;オーバーフロー・ブロック
0
1
2
3
4
5
6
7
8
1-2
2-2
5-2
6-2
1-3
6-3
S1
S2
S3
S4
S5
S6
S7
S8
S9
S10
LC
RPLC
LN
RPLN

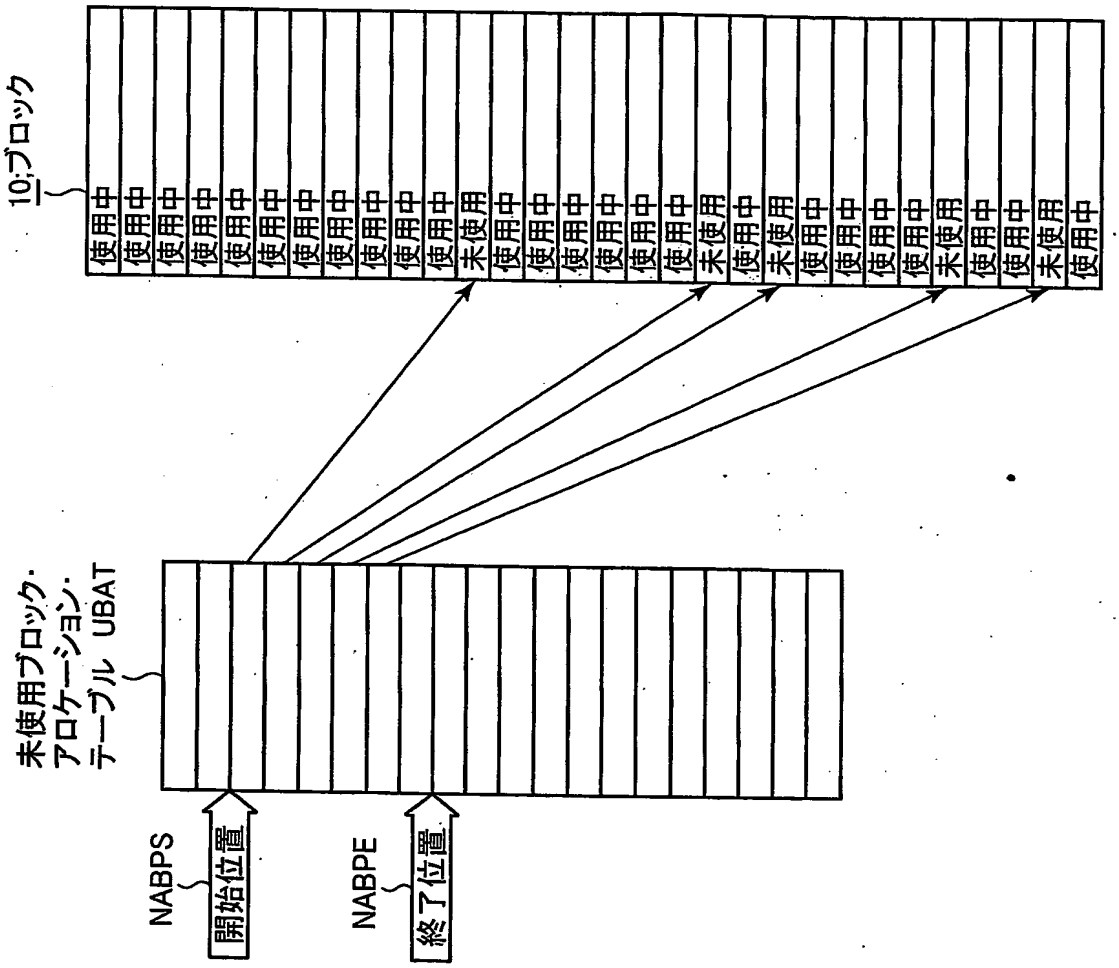
【図４】


1;プライマリー・システム
12;プライマリー・ブロック
13;オーバーフロー・ブロック
14;オーバーフロー・ブロック
10;ブロック
0
1
2
3
4;使用されない
5
6;使用されない
7
8
1-2
2-2
5-2
1-3
S20
S21
S22
S25
S24
S23
LC
0
1
2
3
4
5
6
7
8
9
10
RPLC
排他範囲
LN
0(0)
1(1)
2(1-2)
3(1-3)
4(2)
5(2-2)
6(3)
7(5)
8(5-2)
9
10
11
12
13
14
15
16
RPLN

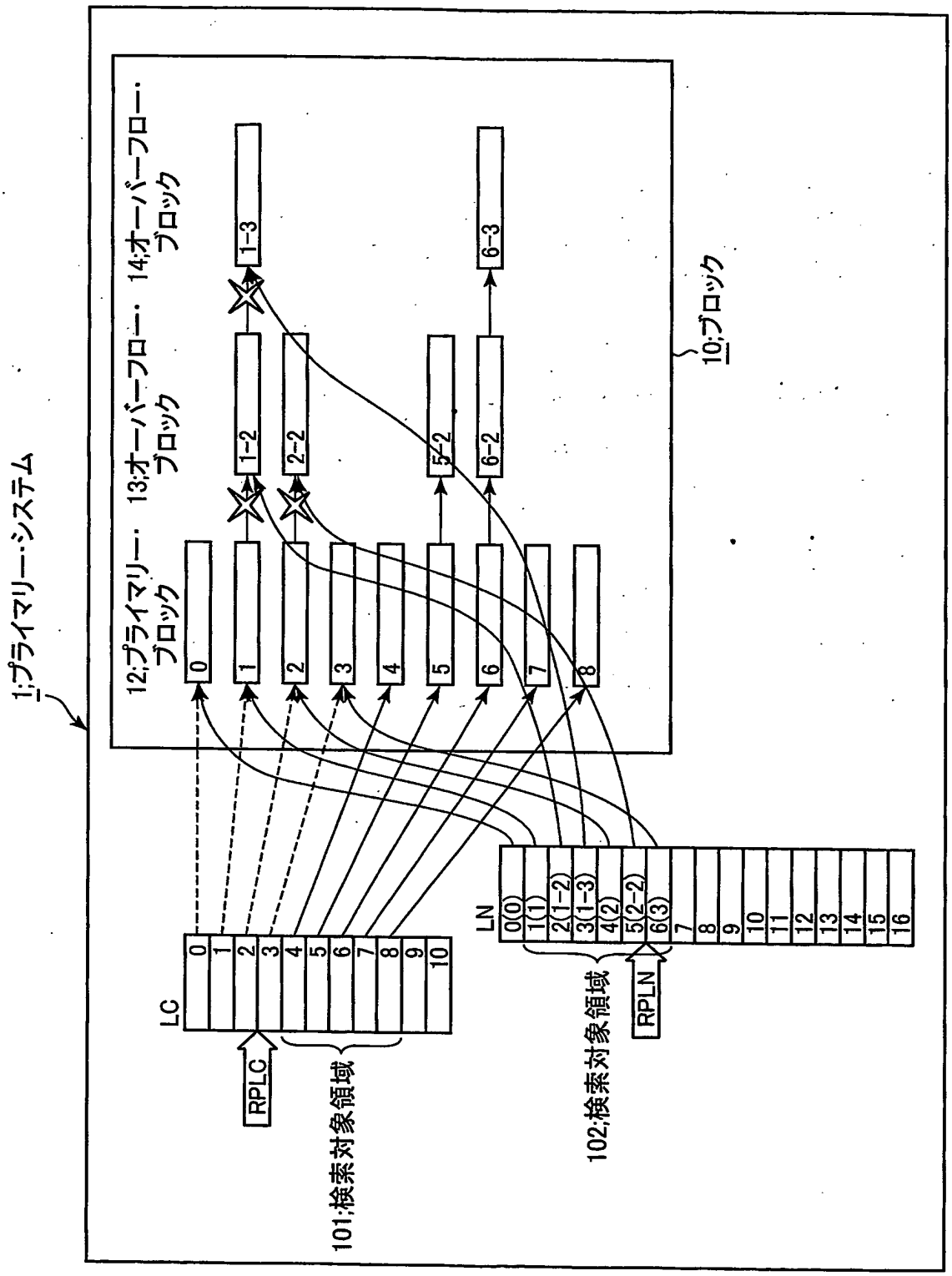
【図５】


10;ブロック
使用中
使用中
使用中
使用中
未使用
未使用
使用中
使用中
使用中
使用中
未使用
使用中
使用中
使用中
使用中
使用中
使用中
未使用
使用中
未使用
使用中
使用中
使用中
使用中
未使用
使用中
使用中
未使用
使用中
未使用ブロック・
アロケーション・
テーブル UBAT
NABPS
開始位置
NABPE
終了位置

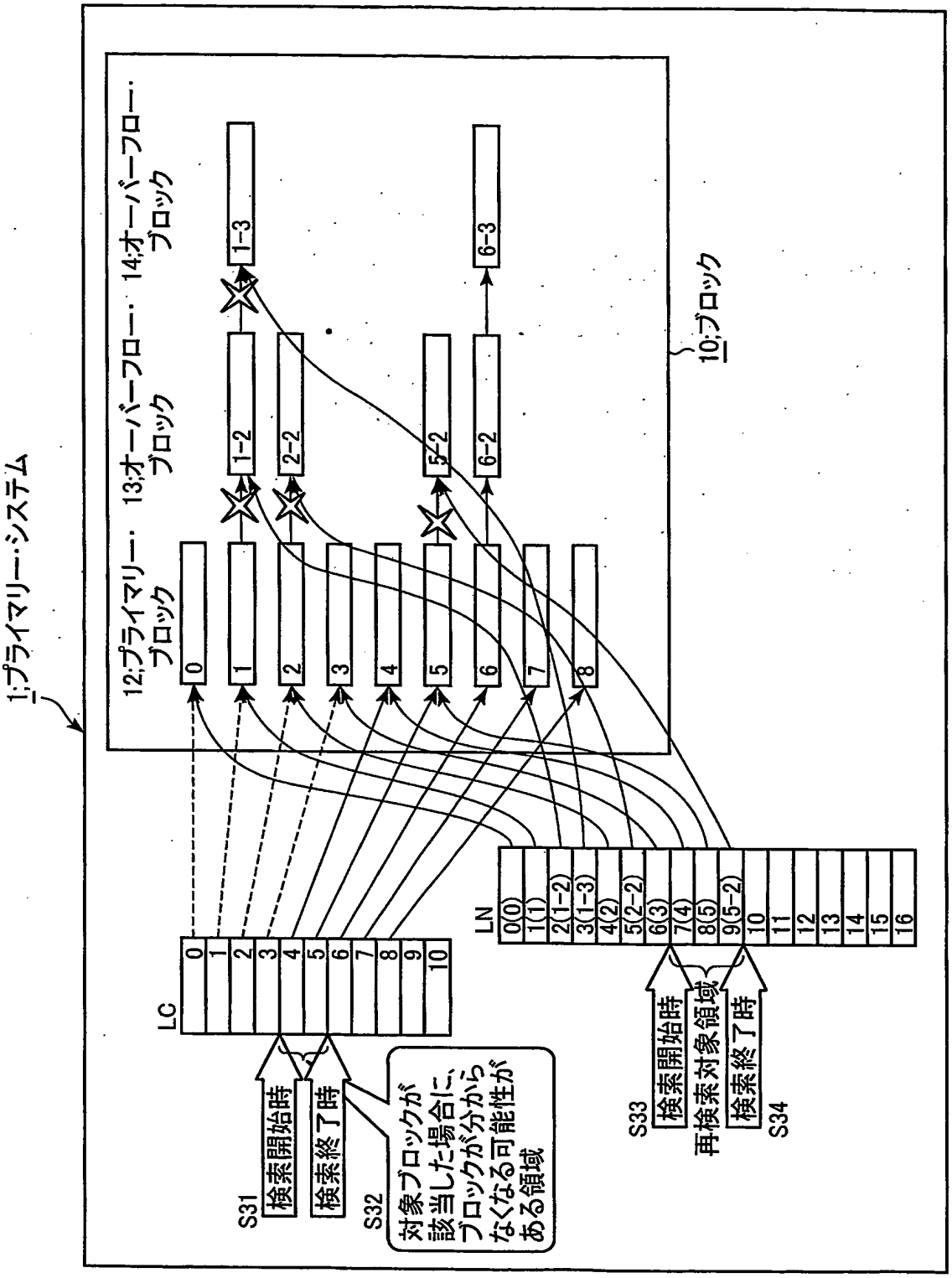
【図６】


10;ブロック
使用中
使用中
使用中
使用中
使用中
使用中
使用中
使用中
使用中
使用中
使用中
未使用
使用中
使用中
使用中
使用中
使用中
使用中
未使用
使用中
未使用
使用中
使用中
使用中
未使用
使用中
使用中
未使用
使用中
未使用ブロック・
アロケーション・
テーブル UBAT
NABPS
開始位置
NABPE
終了位置

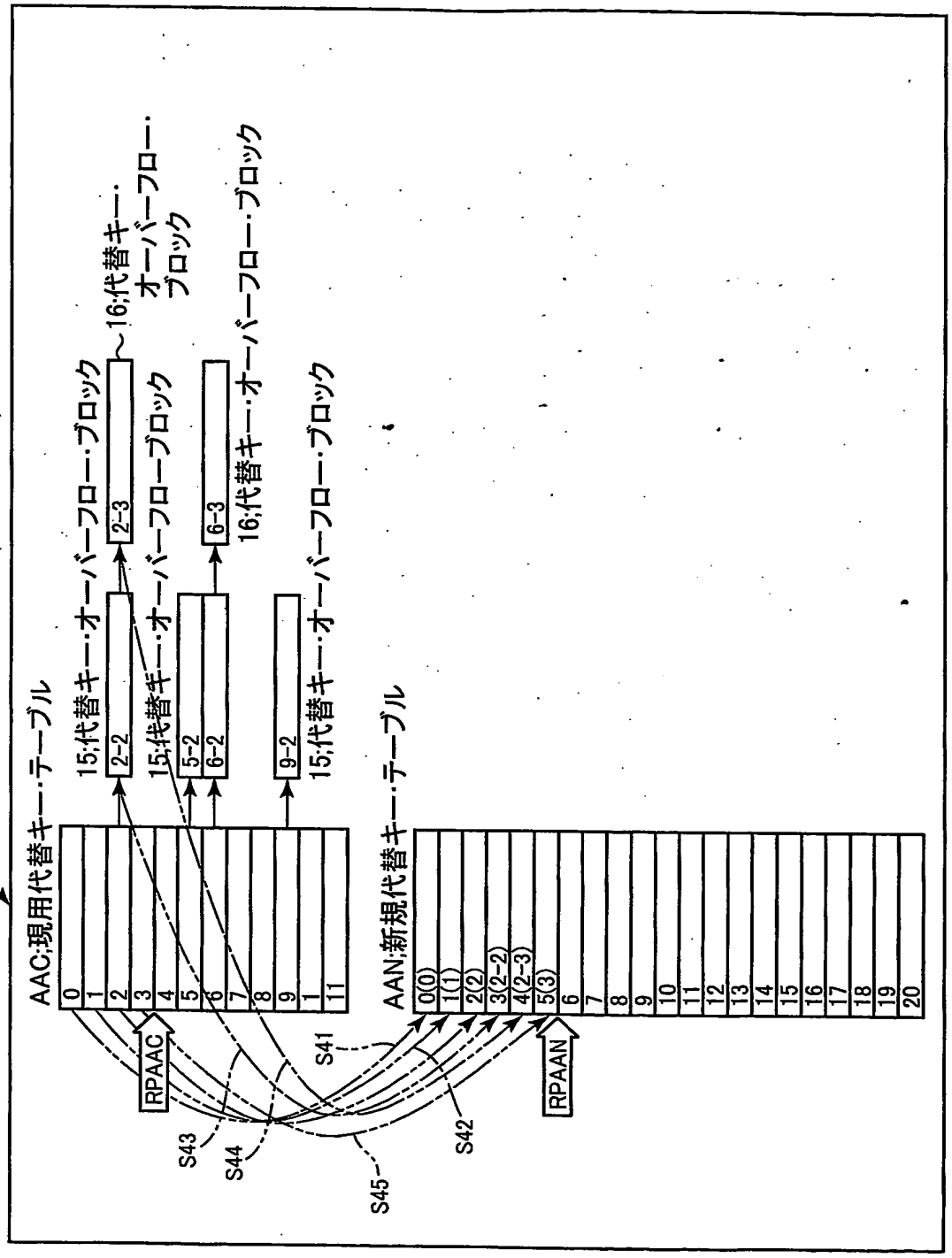
【図７】


1;プライマリー・システム
10;ブロック
12;プライマリー・ブロック
13;オーバーフロー・ブロック
14;オーバーフロー・ブロック
1-2
2-2
5-2
6-2
1-3
6-3
LC
RPLC
101;検索対象領域
LN
RPLN
102;検索対象領域

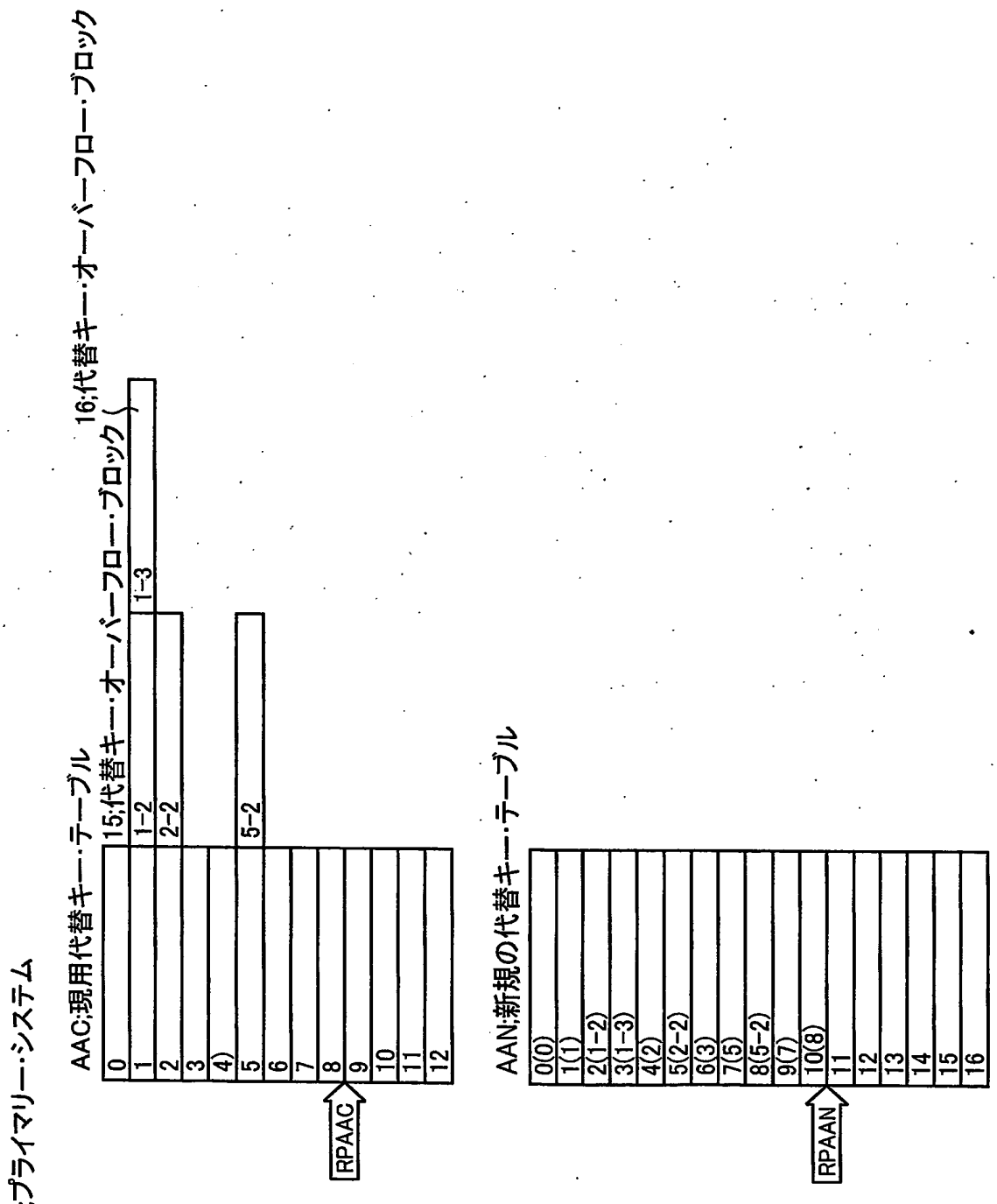
【図８】


1;プライマリー・システム
12;プライマリー・ブロック
13;オーバーフロー・ブロック
14;オーバーフロー・ブロック
10;ブロック
0
1
2
3
4
5
6
7
8
1-2
2-2
5-2
6-2
1-3
6-3
LC
LN
0(0)
1(1)
2(1-2)
3(1-3)
4(2)
5(2-2)
6(3)
7(4)
8(5)
9(5-2)
10
11
12
13
14
15
16
S31
検索開始時
検索終了時
S32
対象ブロックが該当した場合に、ブロックが分からなくなる可能性がある領域
S33
検索開始時
再検索対象領域
検索終了時
S34

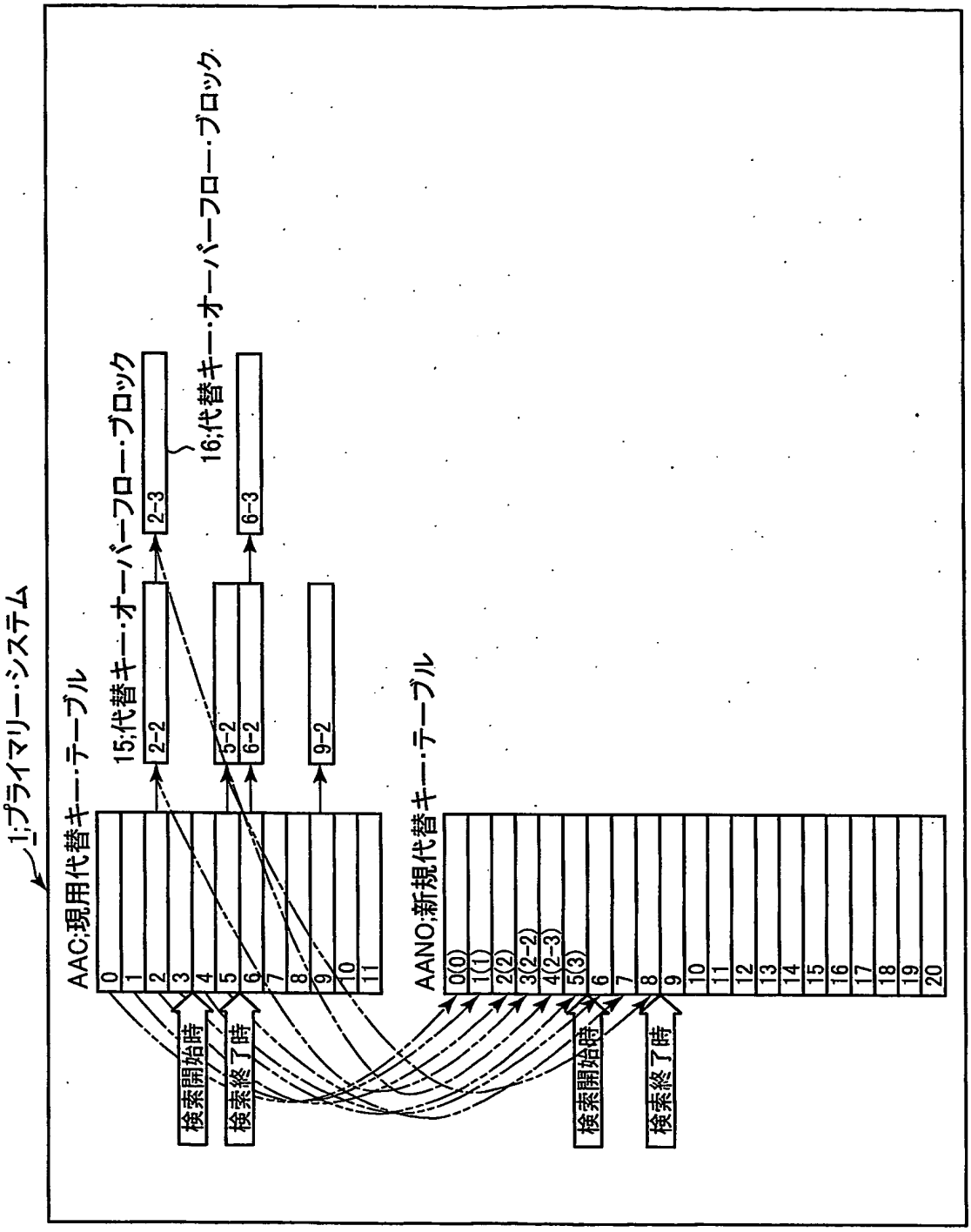
【図９】


1;プライマリー・システム
AAC;現用代替キー・テーブル
0
1
2
3
4
5
6
7
8
9
1
11
15;代替キー・オーバーフロー・ブロック
2-2
16;代替キー・オーバーフロー・ブロック
2-3
15;代替キー・オーバーフローブロック
5-2
6-2
16;代替キー・オーバーフロー・ブロック
6-3
15;代替キー・オーバーフロー・ブロック
9-2
RPAAC
AAN;新規代替キー・テーブル
0(0)
1(1)
2(2)
3(2-2)
4(2-3)
5(3)
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
RPAAN
S41
S42
S43
S44
S45

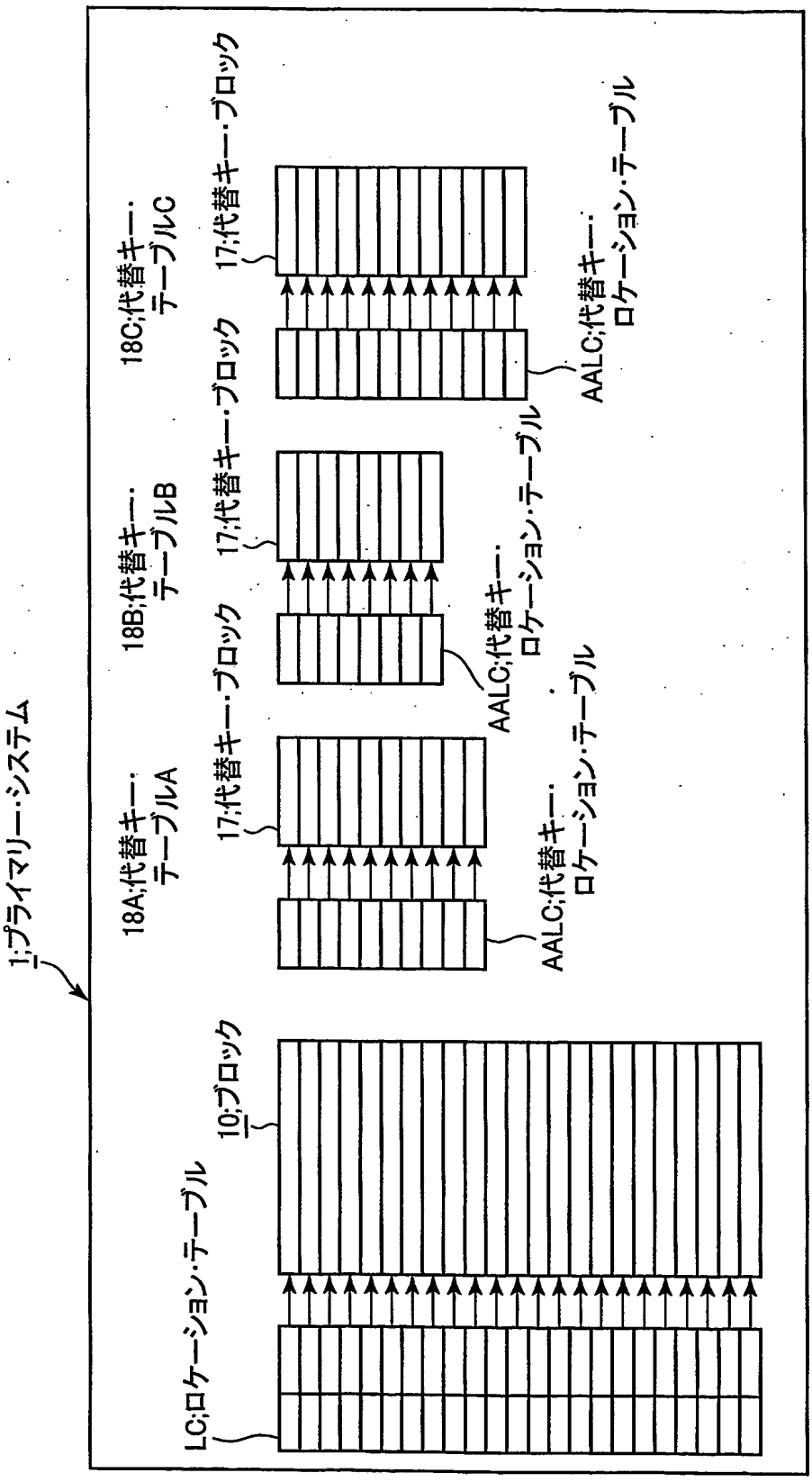
【図１０】


1;プライマリー・システム
AAC;現用代替キー・テーブル
15;代替キー・オーバーフロー・ブロック
16;代替キー・オーバーフロー・ブロック
0
1
2
3
4)
5
6
7
8
9
10
11
12
1-2
2-2
1-3
5-2
RPAAC
AAN;新規の代替キー・テーブル
0(0)
1(1)
2(1-2)
3(1-3)
4(2)
5(2-2)
6(3)
7(5)
8(5-2)
9(7)
10(8)
11
12
13
14
15
16
RPAAN

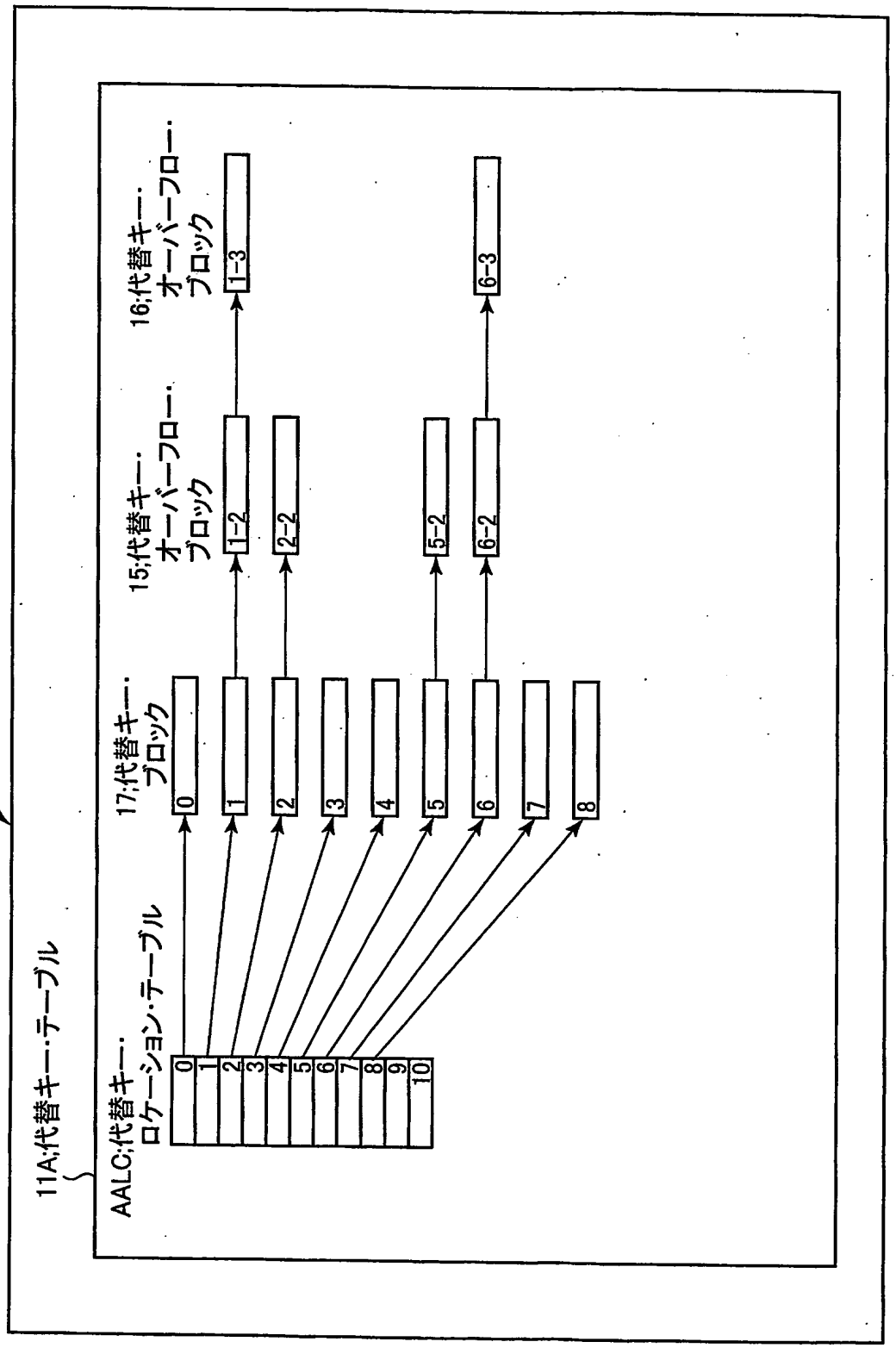
【図１１】


1;プライマリー・システム
AAC;現用代替キー・テーブル
15;代替キー・オーバーフロー・ブロック
16;代替キー・オーバーフロー・ブロック
AANO;新規代替キー・テーブル
検索開始時
検索終了時
検索開始時
検索終了時

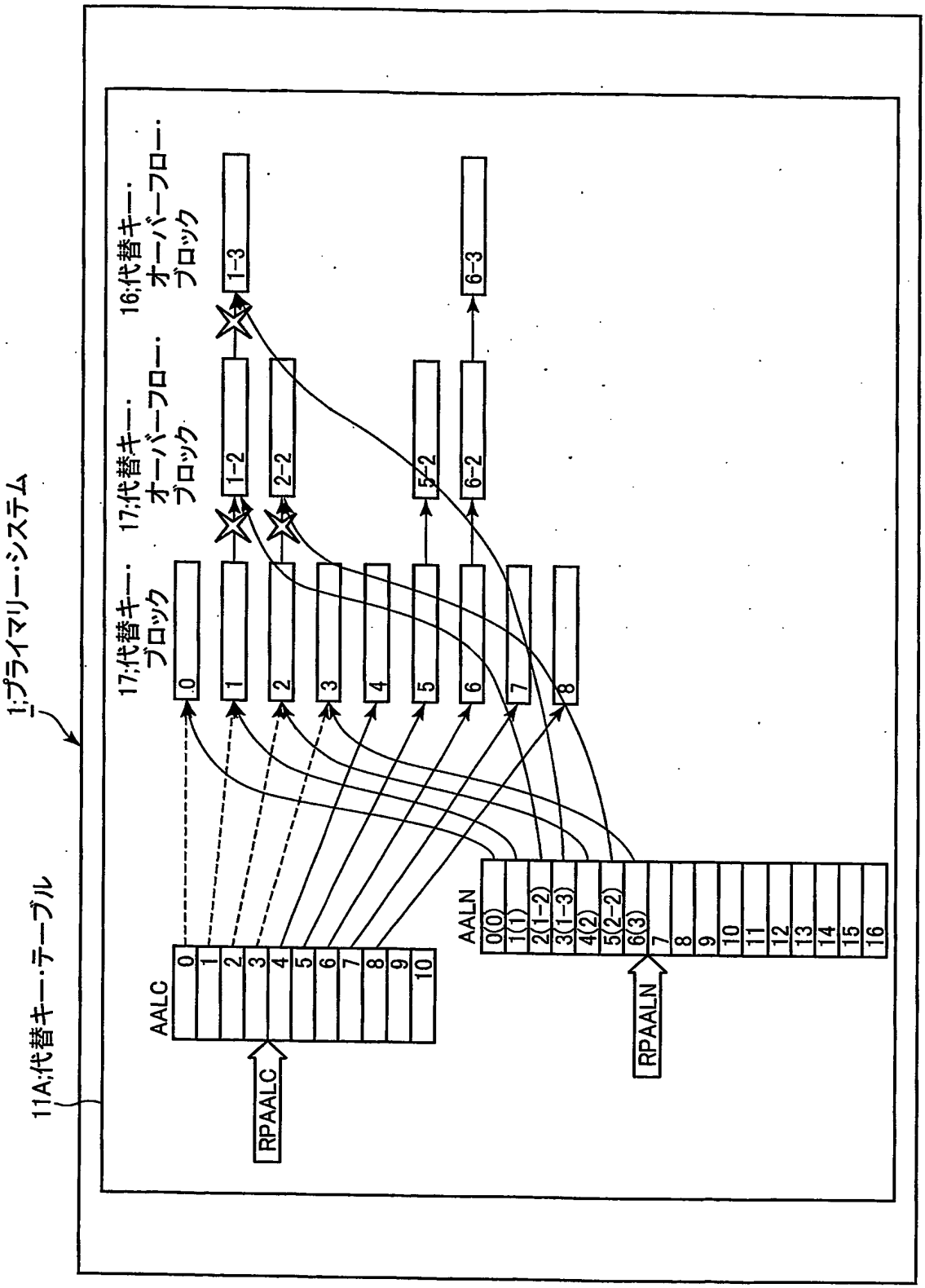
【図１２】


1;プライマリー・システム
18A;代替キー・テーブルA
18B;代替キー・テーブルB
18C;代替キー・テーブルC
17;代替キー・ブロック
17;代替キー・ブロック
17;代替キー・ブロック
AALC;代替キー・ロケーション・テーブル
AALC;代替キー・ロケーション・テーブル
AALC;代替キー・ロケーション・テーブル
10;ブロック
LC;ロケーション・テーブル

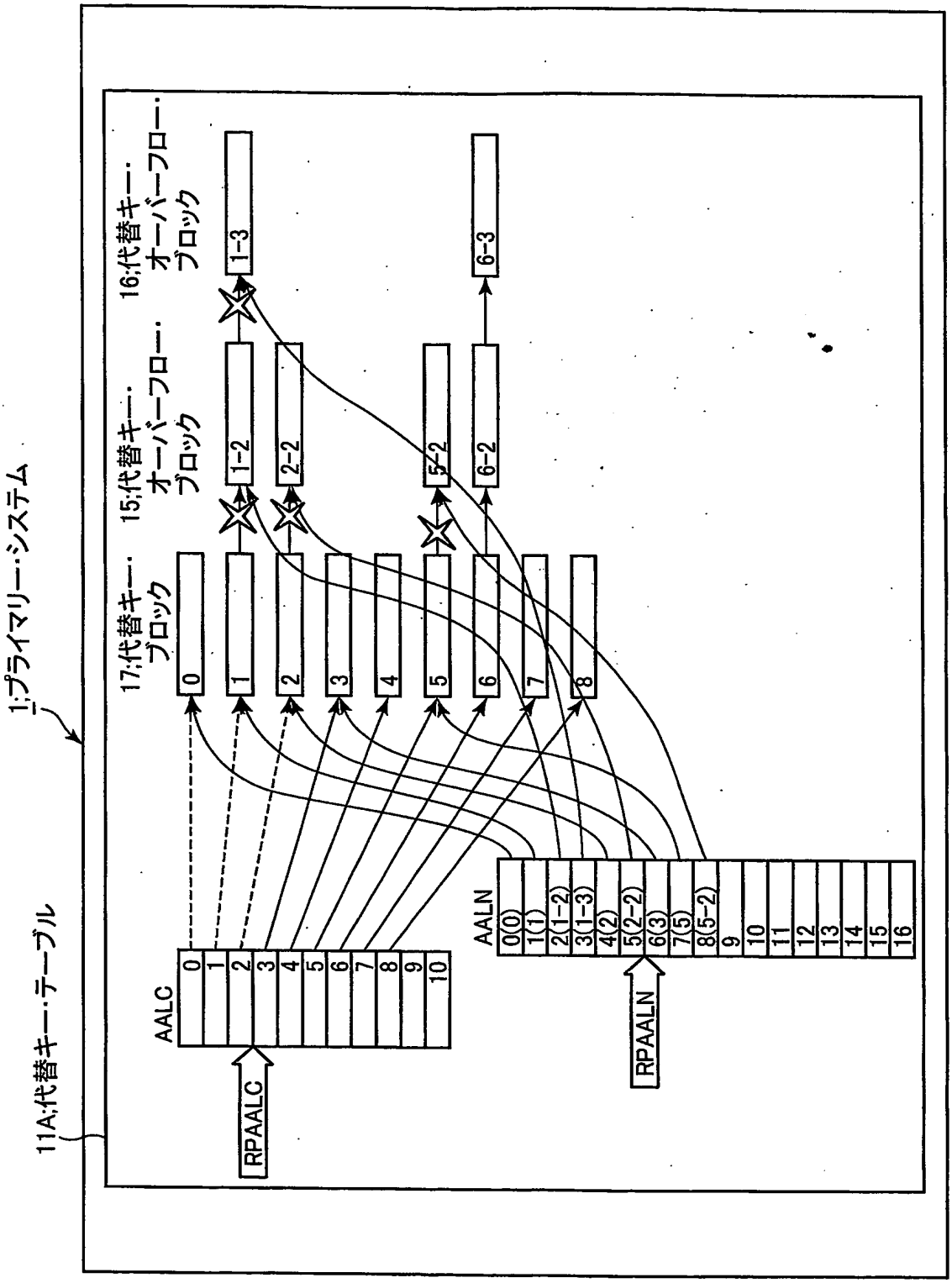
【図１３】


1;プライマリー・システム
11A;代替キー・テーブル
AALC;代替キー・ロケーション・テーブル
17;代替キー・ブロック
15;代替キー・オーバーフロー・ブロック
16;代替キー・オーバーフロー・ブロック
0
1
2
3
4
5
6
7
8
9
10
1-2
2-2
5-2
6-2
1-3
6-3

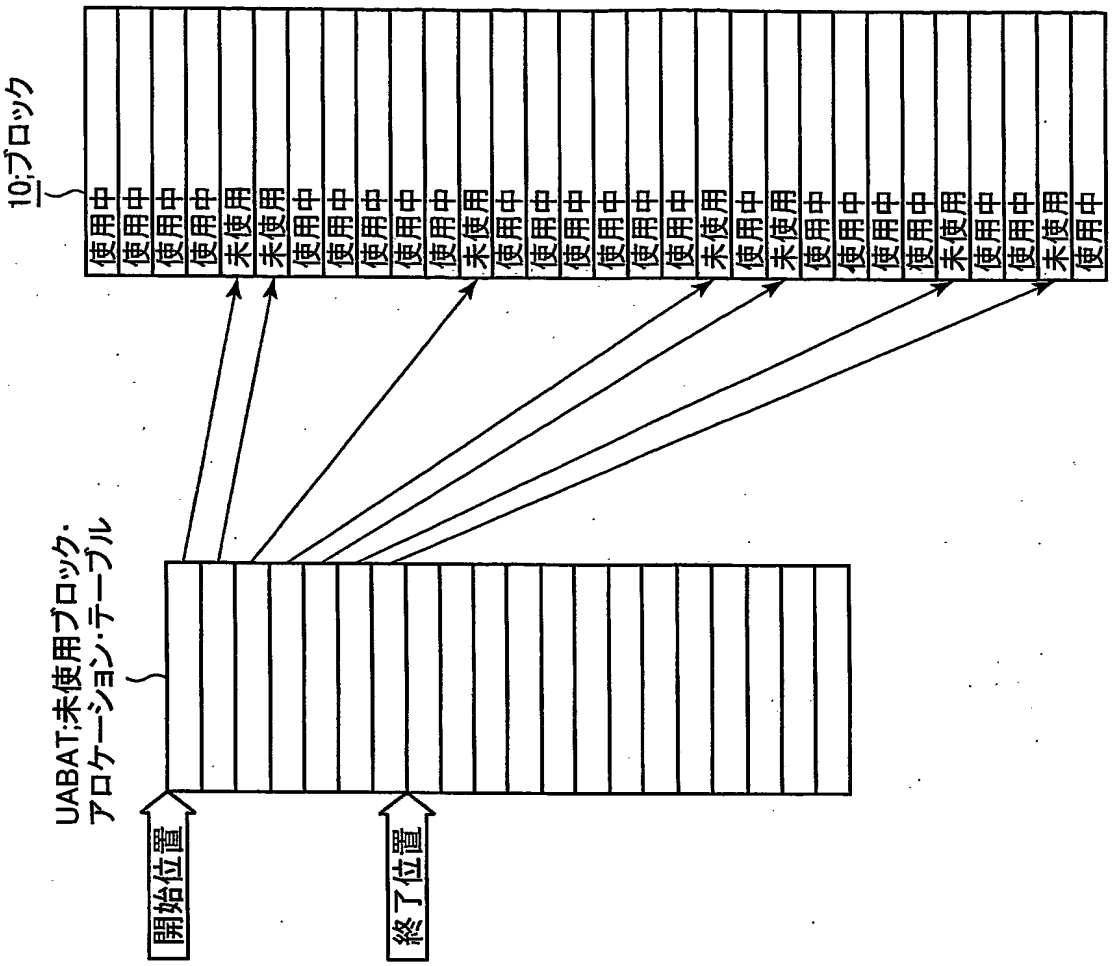
【図１４】


1;プライマリー・システム
11A;代替キー・テーブル
17;代替キー・ブロック
17;代替キー・オーバーフロー・ブロック
16;代替キー・オーバーフロー・ブロック
AALC
RPAALC
AALN
RPAALN
1-3
1-2
2-2
5-2
6-2
6-3

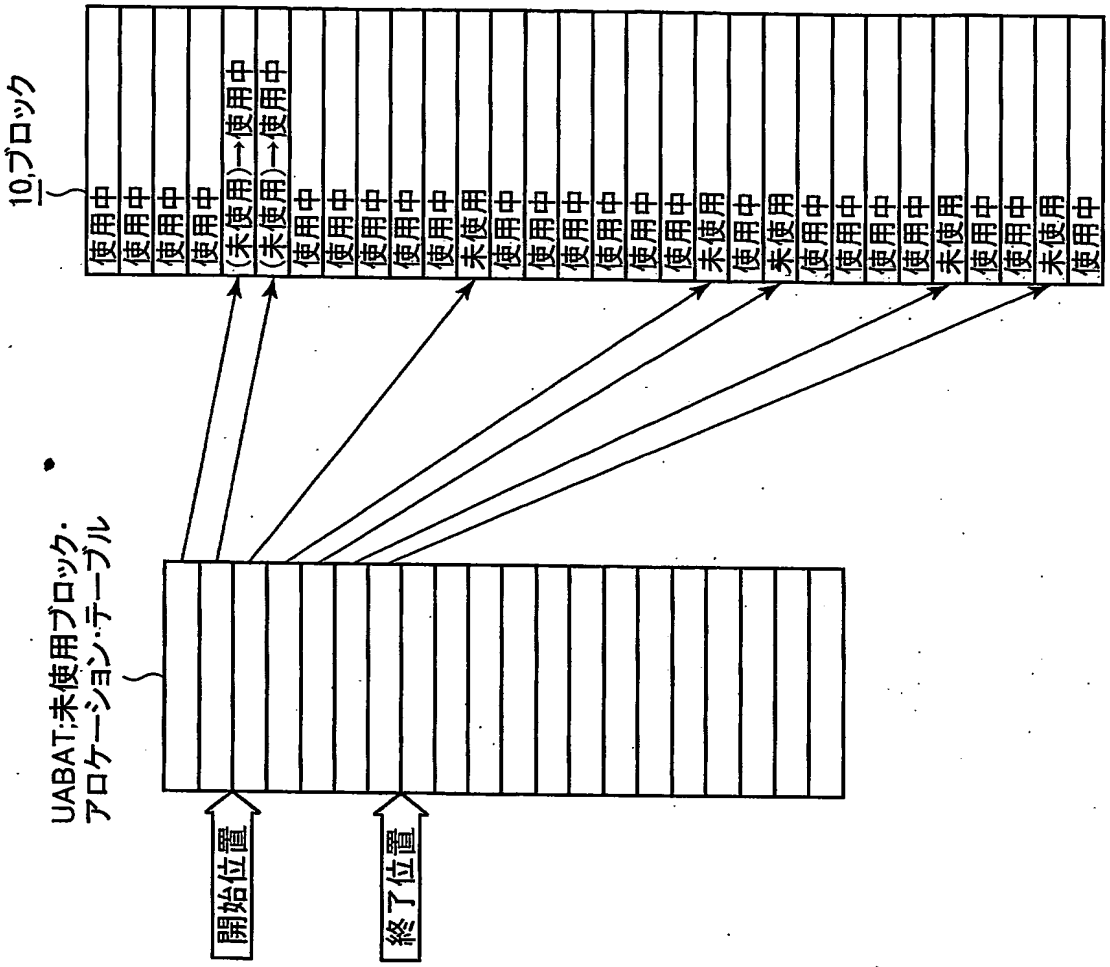
【図１５】


1;プライマリー・システム
11A;代替キー・テーブル
17;代替キー・ブロック
15;代替キー・オーバーフロー・ブロック
16;代替キー・オーバーフロー・ブロック
AALC
RPAALC
AALN
RPAALN
1-2
2-2
5-2
6-2
1-3
6-3

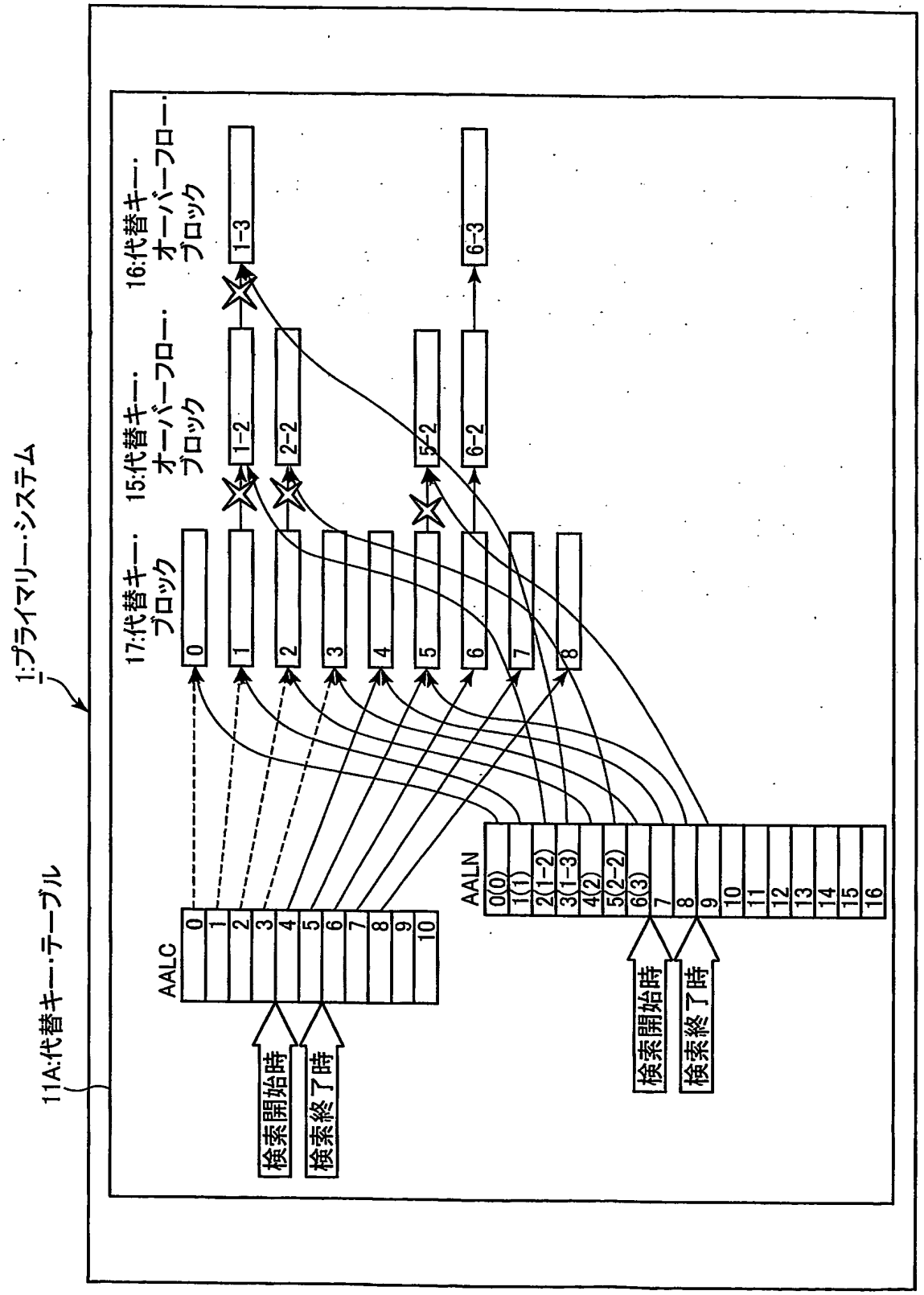
【図１６】


10;ブロック
使用中
使用中
使用中
使用中
未使用
未使用
使用中
使用中
使用中
使用中
使用中
未使用
使用中
使用中
使用中
使用中
使用中
使用中
未使用
使用中
未使用
使用中
使用中
使用中
使用中
未使用
使用中
使用中
未使用
使用中
UABAT;未使用ブロック・
アロケーション・テーブル
開始位置
終了位置

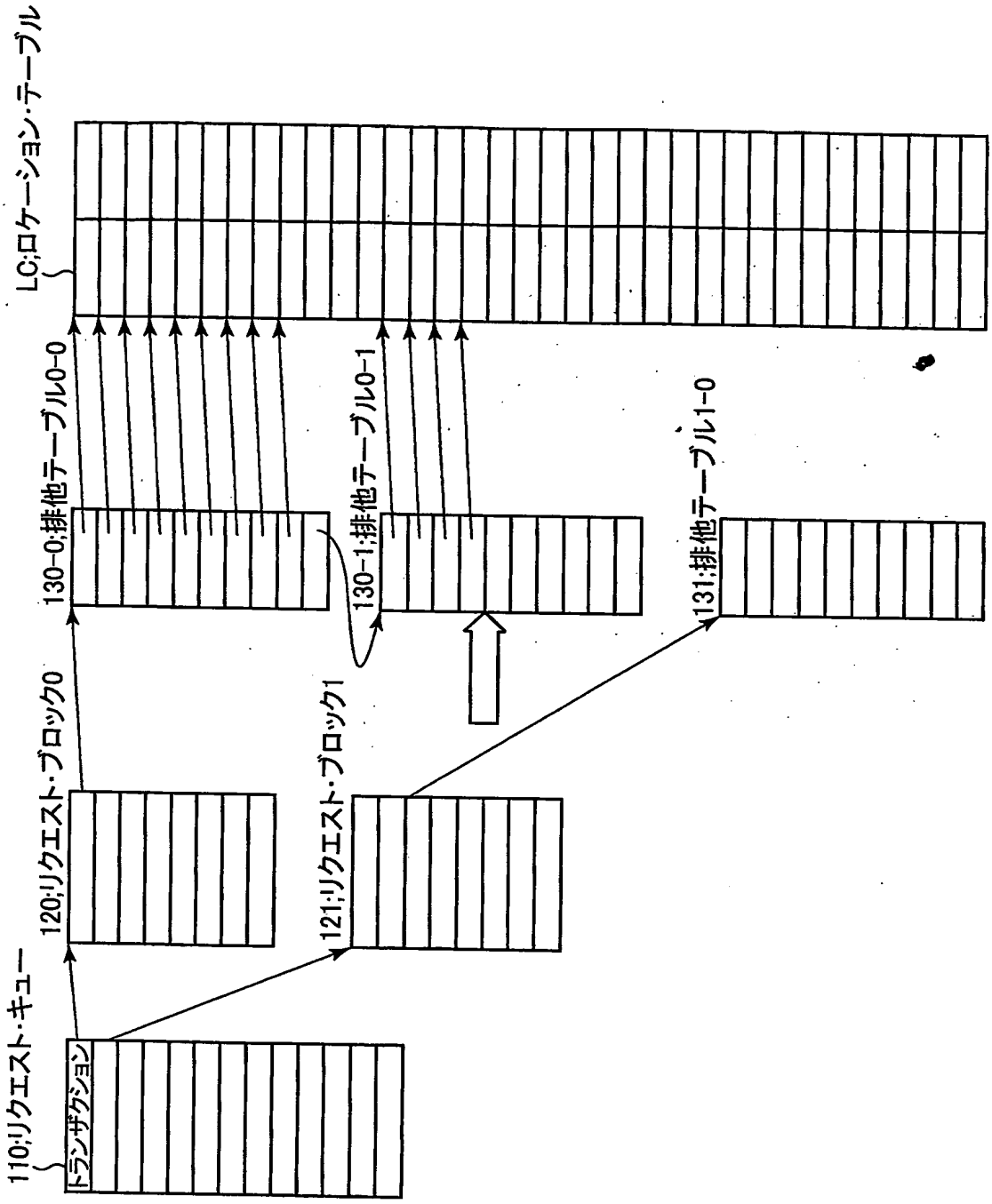
【図１７】


10,ブロック
使用中
使用中
使用中
使用中
(未使用)→使用中
(未使用)→使用中
使用中
使用中
使用中
使用中
使用中
未使用
使用中
使用中
使用中
使用中
使用中
使用中
未使用
使用中
未使用
使用中
使用中
使用中
使用中
未使用
使用中
使用中
未使用
使用中
UABAT;未使用ブロック・
アロケーション・テーブル
開始位置
終了位置

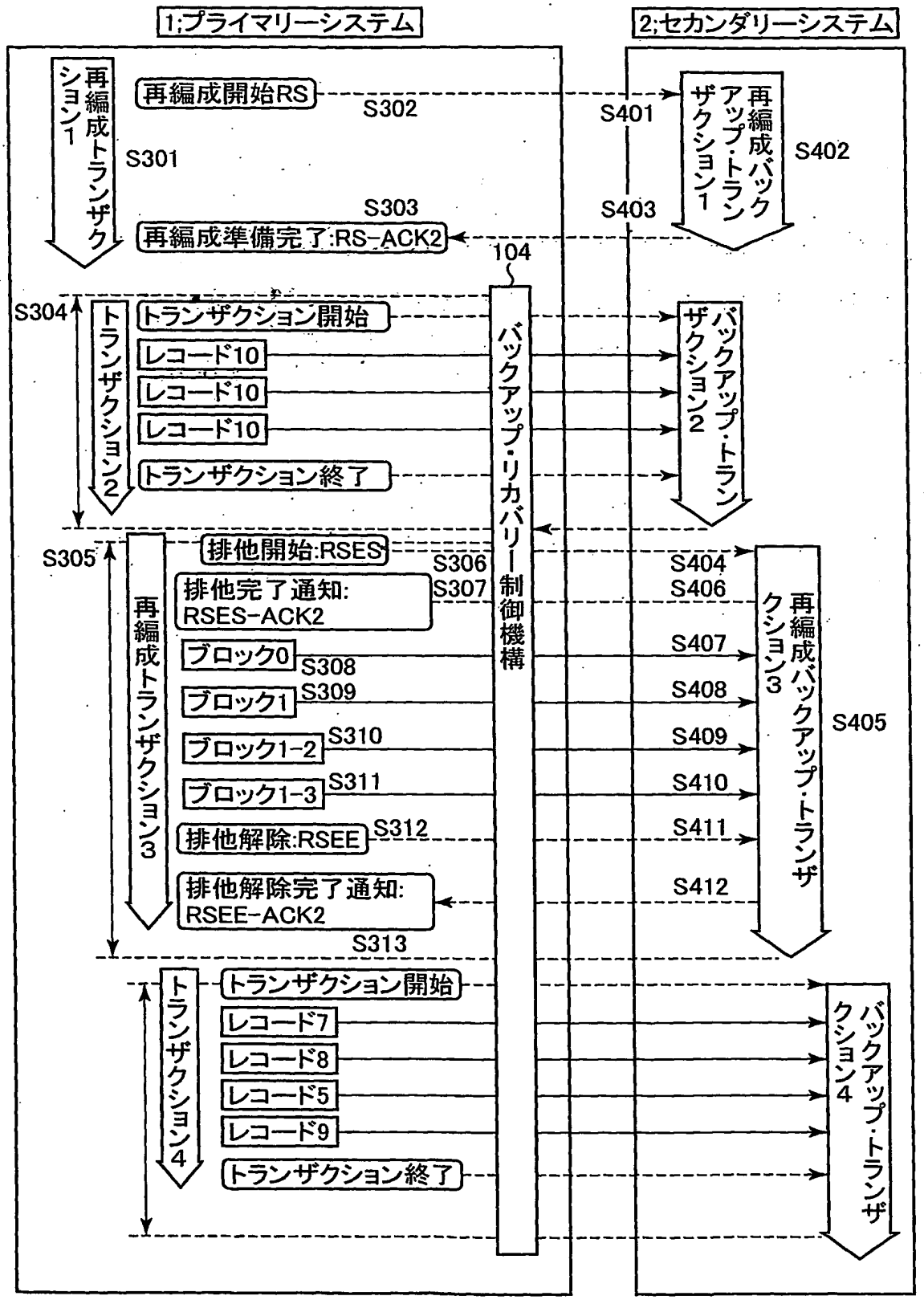
【図１８】


1:プライマリー・システム
11A:代替キー・テーブル
17:代替キー・ブロック
15:代替キー・オーバーフロー・ブロック
16:代替キー・オーバーフロー・ブロック
0
1
2
3
4
5
6
7
8
1-2
2-2
5-2
6-2
1-3
6-3
AALC
0
1
2
3
4
5
6
7
8
9
10
AALN
0(0)
1(1)
2(1-2)
3(1-3)
4(2)
5(2-2)
6(3)
7
8
9
10
11
12
13
14
15
16
検索開始時
検索終了時
検索開始時
検索終了時

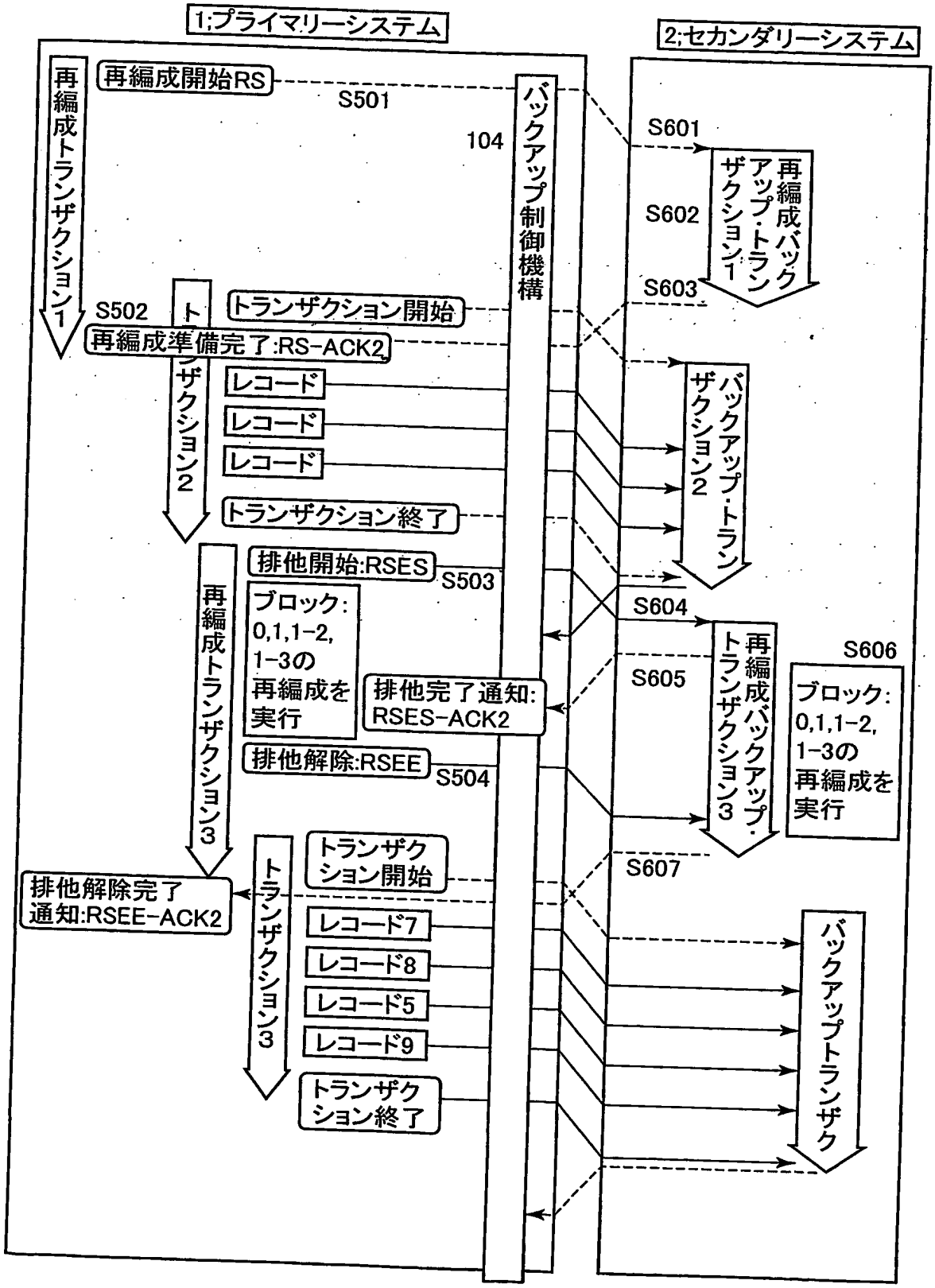
【図１９】


LC;ロケーション・テーブル
130-0;排他テーブル0-0
130-1;排他テーブル0-1
131;排他テーブル1-0
120;リクエスト・ブロック0
121;リクエスト・ブロック1
110;リクエスト・キュー
トランザクション

【図２０】


1;プライマリーシステム
2;セカンダリーシステム
再編成トランザクション1
再編成開始RS
S302
S401
再編成バックアップ・トランザクション1
S402
S301
S303
S403
再編成準備完了:RS-ACK2
104
S304
トランザクション2
トランザクション開始
レコード10
レコード10
レコード10
トランザクション終了
バックアップ・トランザクション2
バックアップ・リカバリー制御機構
S305
再編成トランザクション3
排他開始:RSES
S306
S307
排他完了通知:
RSES-ACK2
S404
S406
S407
ブロック0
S308
S408
ブロック1
S309
S409
ブロック1-2
S310
S410
ブロック1-3
S311
S411
排他解除:RSEE
S312
S412
排他解除完了通知:
RSEE-ACK2
再編成バックアップ・トランザクション3
S405
S313
トランザクション4
トランザクション開始
レコード7
レコード8
レコード5
レコード9
トランザクション終了
バックアップ・トランザクション4

【図２１】


1;プライマリーシステム
2;セカンダリーシステム
再編成トランザクション1
再編成開始RS
S501
104
バックアップ制御機構
S601
再編成バックアップ・トランザクション1
S602
S603
S502
トランザクション開始
再編成準備完了:RS-ACK2
レコード
レコード
レコード
トランザクション終了
バックアップ・トランザクション2
排他開始:RSES
S503
S604
再編成トランザクション3
ブロック:0,1,1-2,1-3の再編成を実行
排他完了通知:RSES-ACK2
S605
再編成バックアップ・トランザクション3
S606
ブロック:0,1,1-2,1-3の再編成を実行
排他解除:RSEE
S504
S607
排他解除完了通知:RSEE-ACK2
トランザクション3
トランザクション開始
レコード7
レコード8
レコード5
レコード9
トランザクション終了
バックアップトランザク

【書類名】　要約書

【要約】

【課題】　特開平１１－３１００９６号公報記載の発明「データ格納検索方式」に対して当該システムを運用したままで再編成を可能とする無停止自動再編成システムの提供。

【解決手段】　現用のロケーション・テーブルＬＣに対し新規のロケーション・テーブルＬＮを設け、単位処理時点で１個または複数個のブロックを対象に、ロケーション・テーブルのエントリーをＬＣからＬＮに書き移すことにより、レコードを格納するブロックと主キーの再編成が可能となる。同様の方法により、代替キー・インデックスの再編成も可能となる。この再編成を連続して実行する。再編成中は、フラットなインデックスの性質を利用し、再編成が何処まで進行したかを示す再編成ポインターを用いて、主キー、代替キーによるデータ検索・追加・更新・削除を実行しながら、データベースを停止せずにデータベースの再編成を可能とする。

【選択図】　図３

特願２００２－２６４２８３

出　願　人　履　歴　情　報

識別番号　　[５９２１５９３２４]

１．変更年月日　　２００１年　８月　３日
[変更理由]　　住所変更
住　所　　東京都多摩市馬引沢２丁目１４番１４号　サンセットヒルズ２
氏　名　　玉津　雅晴

特願２００２－２６４２８３

出　願　人　履　歴　情　報

識別番号　　[５９３０５０５９６]

１．変更年月日　　１９９３年　２月　４日
[変更理由]　　新規登録
住　所　　東京都港区北青山３丁目７番１号
氏　名　　アネックスシステムズ株式会社